Acquisition and On-line Reconstruction of 3D Point Data from Hand-held Laser Scanners and Multi-camera Stereo-matching

vom Fachbereich Informatik
der Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
genehmigte Dissertation

von
Klaus Denker

Wissenschaftliche Aussprache:	10. Juli 2014
Dekan:	Prof. Dr. Klaus Schneider
Promotionskommission:	
Vorsitzender:	Prof. Dr. Markus Nebel
Erster Berichterstatter:	Prof. Dr. Hans Hagen
Zweiter Berichterstatter:	Prof. Dr. Bernd Hamann
Dritter Berichterstatter:	Prof. Dr. Georg Umlauf

D 386

Summary

Three dimensional (3d) point data is used in industry for measurement and reverse engineering. Precise point data is usually acquired with triangulating laser scanners or high precision structured light scanners. Lower precision point data is acquired by real-time structured light devices or by stereo matching with multiple cameras. The basic principle of all these methods is the so-called triangulation of 3d coordinates from *two dimensional* (2d) camera images.

This dissertation contributes a method for multi-camera stereo matching that uses a system of four synchronized cameras. A GPU based stereo matching method is presented to achieve a high quality reconstruction at interactive frame rates. Good depth resolution is achieved by allowing large disparities between the images. A multi level approach on the GPU allows a fast processing of these large disparities.

In reverse engineering, hand-held laser scanners are used for the scanning of complex shaped objects. The operator of the scanner can scan complex regions slower, multiple times, or from multiple angles to achieve a higher point density. Traditionally, *computer aided design* (CAD) geometry is reconstructed in a separate step after the scanning. Errors or missing parts in the scan prevent a successful reconstruction.

The contribution of this dissertation is an on-line algorithm that allows the reconstruction during the scanning of an object. Scanned points are added to the reconstruction and improve it on-line. The operator can detect the areas in the scan where the reconstruction needs additional data.

First, the point data is thinned out using an octree based data structure. Local normals and principal curvatures are estimated for the reduced set of points. These local geometric values are used for segmentation using a region growing approach. Implicit quadrics are fitted to these segments. The canonical form of the quadrics provides the parameters of basic geometric primitives.

An improved approach uses so called *accumulated means* of local geometric properties to perform segmentation and primitive reconstruction in a single step. Local geometric values can be added and removed on-line to these means to get a stable estimate over a complete segment. By estimating the shape of the segment it is decided which local areas are added to a segment. An accumulated score estimates the probability for a segment to belong to a certain type of geometric primitive.

A boundary around the segment is reconstructed using a growing algorithm that ensures that the boundary is closed and avoids self intersections.

Zusammenfassung

Dreidimensionale (3d) Punktdaten werden in der Industrie für Messungen und Reverse Engineering verwendet. Üblicherweise werden solche präzisen Daten mit Laserscannern oder Structured Light Scannern erfasst. Echtzeit Structured Light Kameras und Stereomatching Verfahren liefern Punktdaten niedrigerer Qualität. Das Prinzip hinter all diesen Methoden ist die Triangulierung von 3d Koordinaten aus zweidimensionalen (2d) Kamerabildern.

Diese Dissertation präsentiert eine Methode für Multi-Kamera Stereomatching mit einem System aus vier synchronen Kameras. Mit einer GPU-basierten Stereomatching Methode wird eine hohe Rekonstruktionsqualität bei interaktiven Bildraten erzielt. Große Disparitäten zwischen den Bildern erlauben eine gute Tiefenauflösung. Ein Multi-Level Ansatz erreicht dennoch eine schnelle Verarbeitung.

Handgeführte Laserscanner werden im Reverse Engineering benutzt, um komplex geformte Objekte zu scannen. Der Benutzer des Scanners kann komplexe Regionen langsamer, mehrfach und aus mehreren Winkeln scannen, um eine höherere Punktdichte zu erreichen. Traditionell wird CAD-Geometrie in einem separaten Schritt rekonstruiert. Scanfehler können eine erfolgreiche Rekonstruktion verhindern.

Diese Dissertation stellt einen Online-Algorithmus vor, der die Rekonstruktion während des Scannens ermöglicht. Gescannte Punkte werden zur Rekonstruktion hinzugefügt und verbessern sie online. Der Benutzer kann Bereiche erkennen, an denen die Rekonstruktion zusätzliche Daten benötigt.

Die Punktdaten werden mit einer Octree-basierten Datenstruktur ausgedünnt. Normalen und Hauptkrümmungen werden für die reduzierte Punktmenge berechnet. Diese lokalen Daten erlauben eine Segmentierung per Region Growing. Auf die Segmente werden implizite Quadriken gefittet, deren kanonische Form die Parameter der geometrischen Grundkörper liefert.

Ein verbesserter Ansatz benutzt sogenannte *Akkumulierte Mittelwerte* lokaler geometrischer Daten für eine Segmentierung und Rekonstruktion in einen einzigen Schritt. Um stabile Werte für ein ganzes Segment zu erhalten, werden Daten online zu den Mittelwerten hinzugefügt und entfernt. Die ermittelte Form des Segmentes erlaubt es zu entscheiden, welche Daten hinzugefügt werden. Ein akkumulierter Punktewert schätzt, wie wahrscheinlich es zu einer Art von Grundkörper gehört.

Ein Rand um das Segment wird mit einem Online-Verfahren erzeugt, das sicherstellen soll, dass er geschlossen und frei von Selbstschnitten ist.

Preface

First, I want to thank my advisors, Hans Hagen, Georg Umlauf, and Bernd Hamann for their great support. Especially Georg Umlauf supported me a lot since my diploma studies. He introduced me to the International Research Training Group (IRTG 1131) of Hans Hagen in Kaiserslautern. After the diploma studies, I worked for him at the TU Kaiserslautern and then followed him to the HTWG Konstanz.

The group of Hans Hagen in Kaiserslautern was always a wonderful place to study. Being part of a really international group, the IRTG 1131, was a great experience. It also allowed me to visit many international conferences.

Furthermore, I had the great opportunity to spend several months in the United States at the UC Davis in the group of Bernd Hamann. During our regular meetings at the UC Davis and later via Skype, Bernd Hamann always gave me important advise for my work.

I thank all the nice people I met at the IRTG 1131 in Kaiserslautern, especially Mady Gruys, who is the heart of the group. For sharing great adventures I thank Daniel Burkhart, Daniel Engel, Mathias Hummel, Max Langbein, and Kerstin Müller.

At HTWG Konstanz I also got to know a lot of great people. For their support and influence on my work, I especially thank Manuel Caputo, Daniel Hagel, Jakob Raible, and everyone from the Institut für Optische Systeme Konstanz.

Living over four years in two places, Konstanz and Kaiserslautern, was only possible because of the great support of my family. I thank my mother Herta Denker, my father Helmut Denker, and my brother Peter Denker from all my heart. For the same reason I want to thank my best friends from Kaiserslautern: Sebastian Deußer, Verena Förster, Stefan Kraska, Martin Mainitz, Thomas Schmidt, and Esra Thees. We spent many great evenings playing games and watching movies together.

For their cooperation in the DFG on-line reconstruction project I thank Thomas Schreiber and Ralf Jaumann from Wenzel Knotenpunkt.

The work for this dissertation was supported by AiF ZIM Project KF 2372101SS9, and DFG grants UM 26/5-1 and IRTG 1131.

Contents

CHAPTER 1

Introduction

A growing amount of applications uses scans of *three dimensional* (3d) point data. It is traditionally used in industry for measurements and reverse engineering. Now, with the development of mass market 3d printers, there also is a demand for low-cost 3d scanning solutions.

Precise point data for industrial applications is usually acquired with laser scanners or high precision structured light scanners. For consumer use, there are some real-time structured light devices and there are systems with stereo matching from two cameras. Most of this hardware is not originally intended for 3d scanning. The consumer devices are usually constructed for the recognition of gestures.

All these methods use the same basic principle. They determine 3d coordinates from *two dimensional* (2d) images. Laser scanners project a laser line from a known position into a known direction. A camera mounted in a known distance from this laser captures the light of the laser line reflected by the scanned objects. From the position of the laser light in the camera image the 3d coordinates of points on the laser line are computed. This computation is called *triangulation*.

Similarly structured light scanners project encoded light patterns into space. Traditionally these patterns change over time. Capturing images of all these patterns, a unique address for each pixel of the projector image can be calculated. Triangulating the pixel directions of the projector and the camera allows the computation of 3d coordinates on the lit objects.

Stereo matching is a completely passive reconstruction method, i.e. no additional light sources are used. The image of two or more cameras is compared for similar regions. Using the known external and internal camera parameters, the depth information can be triangulated from the disparity of the same area in multiple camera images. The gained depth information can be transformed to a grid of 3d points.

1.1 Acquisition of 3D Point Data

A method for multi-camera stereo matching is presented in this dissertation. Four synchronized cameras are used to capture the scene. The stereo matching method

is implemented on the *graphics processing unit* (GPU) to achieve a high quality reconstruction at interactive frame rates. High disparities between the camera images allow a good depth resolution. Thus, the cameras are placed at a large relative distance to each other. Fast processing of these large disparities is possible because of a multi level algorithm working directly on the GPU. The images of all four cameras are matched simultaneously to further improve the quality of the reconstruction. Due to a special Y-constellation of the cameras the structure of the epipolar geometry can be used to remedy the effects of the correspondence problem.

The same multi-camera system is used as basis of a hand-held laser scanner. A battery powered hand-held laser line emitter is used to project a laser line on the scanned objects. In contrast to traditional laser scanners, the position of the laser emitter is unknown. The reconstruction is computed by triangulating the image of the laser line in the multiple camera images. This method is very similar to stereo matching. The main difference is that only points on the laser line are processed, instead of the full images, reducing the effects of the correspondence problem.

1.2 On-line reconstruction

For the scanning of complex objects, hand-held laser scanners are used in industry. They are more flexible than stationary 3d scanners. The operator can increase the point density in difficult regions by scanning them slower, multiple times, or from multiple angles. A single scan often contains more than one million data points. Thus it is difficult to see the regions that are not scanned sufficiently. Reverse engineering software usually reconstructs the scan data in a separate step after the scanning. Errors or missing areas in the scan can prevent a successful reconstruction. Later improvement of an existing scan is very difficult.

On-line reconstruction allows the reconstruction of CAD geometry during the scanning of an object. The reconstruction is improved on-line by adding point data from the laser scanner. Areas that need improvement can be detected by the operator while scanning.

Two different approaches to on-line reconstruction of geometric primitives are presented in this dissertation. The first approach uses separate steps for segmentation and reconstruction. An octree based data structure is used to thin out the point data from the laser scanner. Then local normals and principal curvatures are estimated for the reduced points. The segmentation uses a region growing approach on these local geometric values to group the reduced point set to segments. It aims at dividing the point set into segments that belong to the same geometric primitive. In a second step, the geometry of each segment is reconstructed. An implicit quadric is fitted to each segment. The canonical form of the quadric yields the parameters of the reconstructed geometric primitives. By this approach, planes, cylinders, cones, and spheres can be reconstructed.

The second approach does not use local data for region growing. Instead so called *accumulated means* of the local geometric properties are computed. This allows a

much more robust reconstruction. The accumulated means not only provide scores for the segmentation. They also allow the direct computation of the parameters of the reconstructed geometric primitives. No additional reconstruction or fitting steps are necessary.

For rendering of the reconstructed geometry, it is necessary to reconstruct a boundary. A on-line boundary reconstruction is presented, that reconstructs a closed polygonal boundary and tries to avoid self intersections. Because 2d surfaces in 3d space can have multiple boundaries, the boundary reconstruction supports splitting and merging of boundaries during on-line reconstruction. The reconstructed boundary polygons are used to trim the reconstructed segments of geometric primitives for rendering.

CHAPTER 2

Fundamentals

Scan data can be acquired in multiple ways. This chapter gives an overview of these methods.

Laser scanners emit a line of laser light to the scene and reconstruct the coordinates of the points on this line. Usually laser scanners have the advantage of a much higher precision than the other discussed methods. They can only reconstruct a single laser line at a time (Teutsch, 2007).

Structured light methods reconstruct larger regions at a time. A set of light patterns is projected onto the scene. The depth information is reconstructed from the displacement of these light patterns for known camera positions (Posdamer and Altschuler, 1982; Frankowski et al., 2000; Zhang and Yau, 2006). Most of these methods work with visible light and require a dark environment without interfering light sources.

To reconstruct depth information without a light source, stereo matching approaches like Furukawa and Ponce (2007); Klaus et al. (2006); Scharstein and Szeliski (2002); Yang and Pollefeys (2005); Denker and Umlauf (2011a) are used. They use a system of two or more cameras with known positions to reconstruct depth information from the images. Similar image regions in multiple images are detected. The displacement of their position allows to reconstruct the depth. The detection of similar regions is prone to systematic errors from light conditions, reflections, and repetitions in the images.

2.1 Laser Scanners

Depending on the scale of the scanned objects, most laser scanners use one of two basic principles (Jarvis, 1983a). Large objects or landscapes are measured with *time-of-flight* (TOF) scanners. Small objects that need a high precision are scanned with *triangulation scanners*.

TOF scanners emit a laser pulse that travels from the emitter to the scene and back to a camera. The time this impulse needs to cover this distance is measured. Because the speed of light is very high, this method is able to cover large measuring

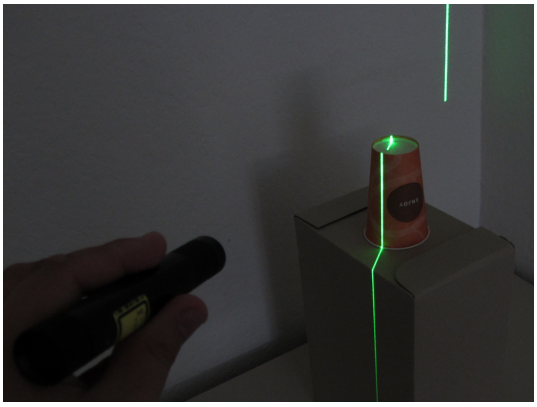


Figure 2.1: Laser line projected by a line-laser probe.

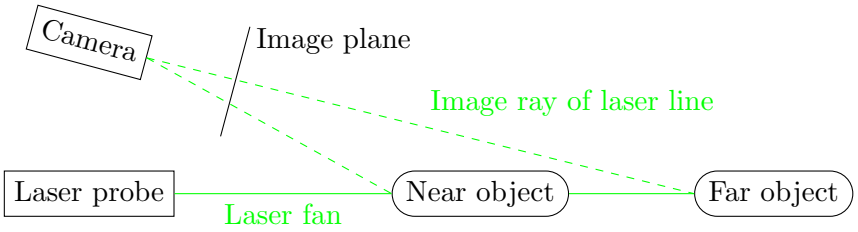


Figure 2.2: Principle of a triangulation scanner.

distances. Therefore, TOF scanners are used for range sensing in robotics (Jarvis, 1983b) and airborne 3d scanning in geo sciences (Wehr and Lohr, 1999).

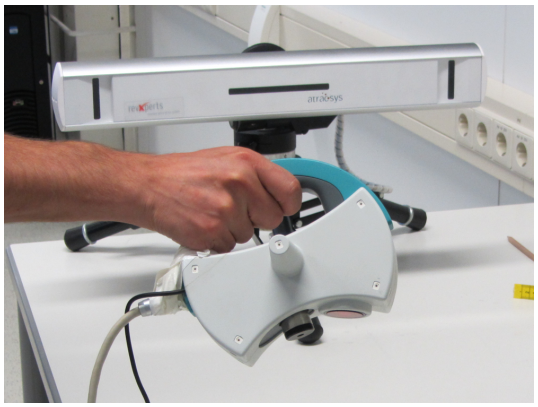
Triangulation scanners are limited to short distances and allow a very high precision. A camera is placed in a known distance to a line-laser emitter and measures the displacements of the laser line. They are used in engineering for quality measurements and reverse engineering (Teutsch, 2007).

2.1.1 Triangulation laser scanners

Triangulation laser scanners usually consist of a *line-laser probe* and a *camera* that are mounted in a fixed distance and angle to each other. A fan of laser light is projected to the scanned objects. Perpendicular to the laser fan, the camera is mounted in a fixed distance to the line-laser probe. Depending on the distance of the scanned objects, the laser-line is displaced. In Figure 2.1 the points of the laser line on near objects are displaced towards the bottom left. Figure 2.2 illustrates how the distance of objects changes the picture of the laser line on the image plane. The far object has an image at the center of the image plane while the image of the near object is displaced to the bottom.

A point of the laser line can be transformed into 3d coordinates by constructing a ray from the camera through the according pixel in the image plane, see Figure 2.2.

(a) Tracking with a measurement arm.



(b) Tracking with an optical tracking system.

Figure 2.3: Two hand-held laser scanners.

This ray is intersected with the plane spanned by the laser fan to get 3d coordinates. For these calculations it is necessary to know all camera parameters and the relative position and orientation of the line-laser probe to the camera. The gained coordinates are relative to the position and orientation of the line-laser probe or the camera.

An early example of such a triangulation laser scanner is presented in (Agin and Binford, 1976). Moving mirrors are used to move the laser fan over the scanned objects. With the resulting lines of 3d coordinates the objects are reconstructed from generalized cylinders.

The triangulation scanner in Shirai and Tsuji (1972) uses light that is projected through a slit to the scene. This way, a fan of light similar to the one of a line-laser is created. The line is reconstructed using multiple image filter operations.

2.1.2 Hand-held laser scanners

Hand-held laser scanners are small triangulation scanners that are moved by an operator by hand. The operator holds the scanner device in his hand and moves it over the scanned object's surface. This allows a greater flexibility than mounted laser scanners that are moved in fixed steps. Regions of the object that are difficult to capture can be scanned slower to achieve a higher point density. It is also possible to scan the same region from multiple directions.

A hand-held laser scanner system always needs to know the exact position and

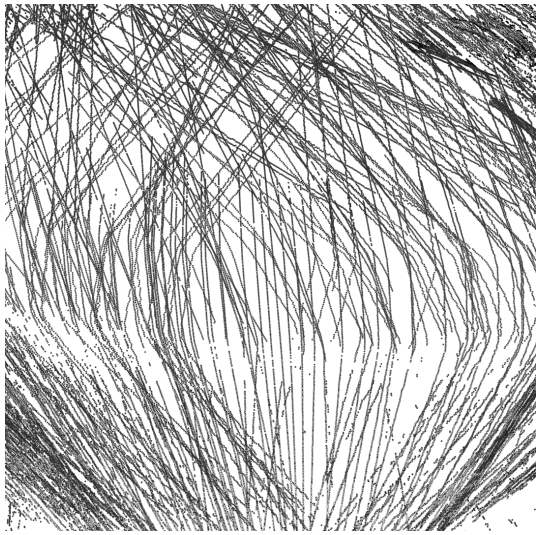


Figure 2.4: Scan data from an hand-held laser scanner. The 3d coordinates are structured in scan lines.

orientation of the laser scanner device. This *tracking* is usually done by a measurement arm, see Figure 2.3a, or by an optical tracking system using markers and a multi-camera system, see Figure 2.3b.

The data from these scanners is organized in so called *scan lines*. A scan line is a line of 3d coordinates that is scanned at the same time. The data a hand-held laser scanner produces during scanning is a stream of such scan lines. In Figure 2.4 the point cloud of an hand-held laser scanner is displayed.

2.1.3 Low-cost laser scanners

High precision time measurements are require for TOF scanners (Jarvis, 1983a). Therefore, it is not possible to build a low cost scanner using this technique.

Low cost laser scanners usually are triangulation scanners. In Winkelbach et al. (2006) a web-cam and a line-laser probe are used. Because the positions of the laser probe and the camera are unknown, cardboard sheets with a known background pattern are used for calibration and triangulation. Based on the parts of the laser line that are visible on the sheets, the plane of the laser fan is estimated. Together with the camera position relative to the sheets, this information is used to compute the coordinates of the whole laser line.

A similar approach was used in Bender et al. (2012) to build a triangulation system from an already available multi-camera system. The multi-camera system allows a reconstruction of the laser line without knowing the position of the laser emitter. No cardboard sheets are required. A more detailed description of this method is provided in Section 3.3.

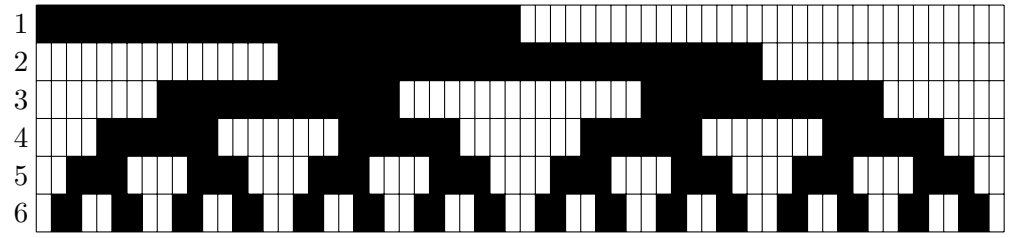


Figure 2.5: First 6 pictures of a horizontal gray code pattern.

2.2 Structured Light

A laser scanner can only capture one line of 3d coordinates at each timestep. Structured light scanners can capture a whole area at once from a small sequence of pictures. They project multiple coded patterns to the scene and read these patterns from a different angle using a camera. Similar to laser scanners a triangulation between the recognized pixels in the camera image and the projector image is done to reconstruct the 3d coordinates.

An overview of different structured light methods is provided in Sá et al. (2002).

2.2.1 Classic approaches

Classic structured light approaches use a set of binary patterns that are emitted by a projector. Early approaches project a grid of laser points that can be masked individually by a shutter (Posdamer and Altschuler, 1982). The horizontal and vertical position of a laser dot is binary encoded to a unique address. Using the shutter, the laser dot is masked if the according bit of the binary code is not set. From an initial image containing all dots and multiple images with the encoded coordinates, it is possible to reconstruct the exact grid coordinates for each laser dot visible from the camera.

Later, slide projectors and LCD projectors were used for structured light reconstruction. This causes some problems at the boundaries between neighboring stripes. Bits from both stripes may be mixed, resulting in large positional errors. To avoid these problems, Gray code patterns were used (Inokuchi et al., 1984; Sansoni et al., 1997). Figure 2.5 shows a example encoding of the horizontal coordinate. In a gray coded image sequence, neighboring values differ at most in a single bit. This ensures that the position error of misinterpreted encodings near the boundaries between stripes is at most the size of one pixel, if the camera resolution is high enough.

2.2.2 Real time approaches

Early real-time structured light approaches use single images with colored sinus patterns (Tajima and Iwakawa, 1990). A diffraction grating is used to spread white light into a fan of rainbow colored light. Because the positions of the camera and the

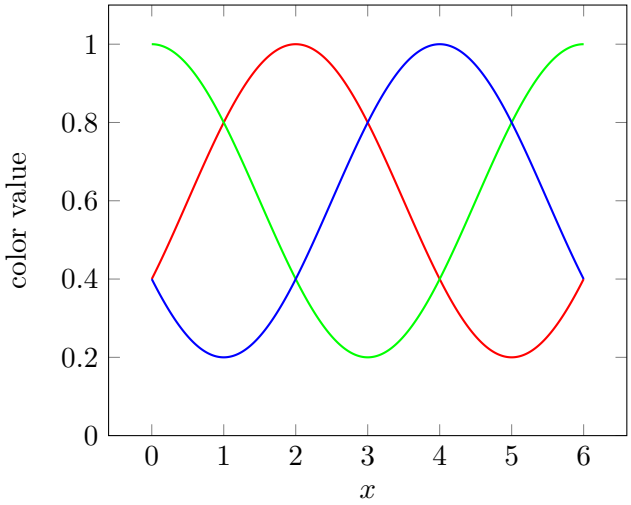


Figure 2.6: Three phase shifted sinus patterns (Zhang and Yau, 2006).

projector only differ horizontally, it is sufficient to reconstruct the horizontal image coordinates. A monochrome CCD camera is used with two color filters. From the color values, the position in the spectrum is computed.

Digital light processing (DLP) projectors use *digital micromirror devices* (DMD) to generate high frequency monochrome images. A wheel of color filters partitions the image sequence to the primary colors. In Frankowski et al. (2000) a DMD with a frame rate of 360 Hz is used without a color wheel to project a fast sequence of sinus stripes. Multiple images from a phase-shifted sinus function of the same frequency and amplitude are used to determine the phase angle for each pixel, see Figure 2.6. Combining the phase angles of multiple sets of pictures with different frequencies allows to reconstruct a wider depth range.

In Zhang and Yau (2006) a similar system is used. An inexpensive DLP projector is modified by removing the color wheel. Three images of a sinus pattern with shifted phase-angles are combined to the three color channels of the input image, see Figure 2.6. Without further modifications, the projector displays a loop of these three images with a frame rate of 180 Hz. Again, the depth reconstruction is done by calculating the phase angle from the shifted sinus patterns. In addition to a fast synchronized monochrome camera, the image of a slower color camera is used to apply a color texture to the reconstructed shapes.

2.2.3 Encoded patterns

A structured light reconstruction from a single image can be achieved by projecting encoded patterns to the scene. An early example of this technique is Vuylsteke and Oosterlinck (1990). The image consists of a checkerboard pattern with data encoded into markers at the corners of the squares, see Figure 2.7. The local area of a pixel

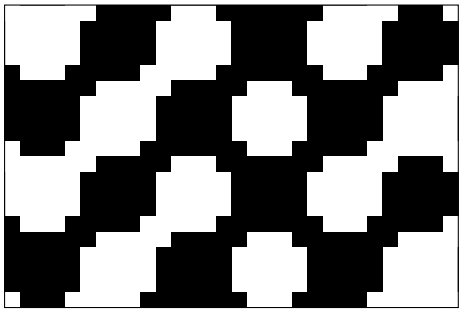


Figure 2.7: Encoded checkerboard pattern for structured light reconstruction (Vuylsteke and Oosterlinck, 1990).

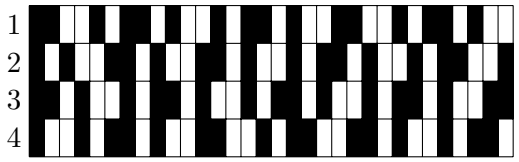


Figure 2.8: Loop of four encoded patterns (Rusinkiewicz et al., 2002).

contains a unique code of 6 bit that encodes its horizontal position. Encoding of the bit pattern with pseudonoise sequences ensures that each pattern is unique while still achieving the optimal length.

In Rusinkiewicz et al. (2002) a loop of four patterns with a special encoding is projected sequentially, see Figure 2.8. Tracking a stripe boundary in four consecutive camera pictures allows to read the value of two stripes in four time steps. Thus, each tracked boundary edge provides a unique code of eight bit. Again, this code is used to identify the horizontal position in the projector image.

The Microsoft Kinect device uses a structured light approach. A small infrared

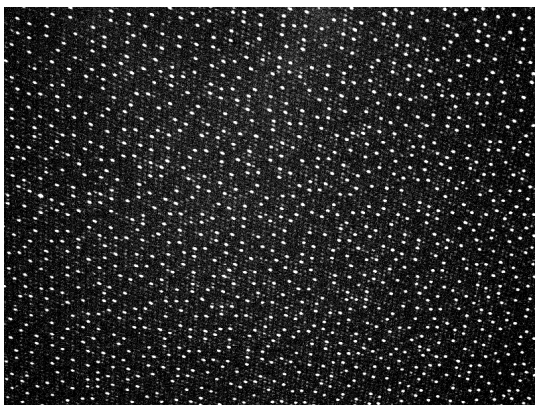


Figure 2.9: Infrared photograph of the dot pattern of a Microsoft Kinect device.

projector casts a static pattern of dots to the scene, see Figure 2.9. A high resolution infrared camera takes an image of this pattern from a horizontally shifted position. The processing of the camera picture to a depth map is done on a special purpose microprocessor within the Kinect. Because of the low price and the high framerate the Microsoft Kinect is often used for gesture recognition (Caputo et al., 2012).

2.3 Stereo matching

So far, only active methods for scan data acquisition were presented. A laser or projector is used to illuminate the scanned object. To reconstruct the shape of an object without using an active light source, stereo matching is used. The depth information is computed from two or more 2d camera images by matching similar regions of the images. Applications for stereo matching range from remote sensing to robotics, archeology, cultural heritage, reverse engineering, and 3d face recognition (Murray and Little, 2000; Pham and Hieu, 2008; Koch et al., 1999; Voltolini et al., 2007). The main advantage of stereo matching is that only natural light is used for the data acquisition. There is no artificial interaction with the scanned object that might do any harm.

An important challenge of stereo matching is the trade-off between the quality of the depth map and the time to compute the depth map. For some applications a real-time computation is not important. Therefore, many stereo- and multi-view-matching methods focus on high quality results instead of fast computation times. These high quality methods need at least several seconds to compute a single depth map from one set of images (Klaus et al., 2006). However, for robotics faster computation times are more important than the quality of the depth map. This led to the development of GPU based real-time matching methods (Yang et al., 2002; Yang and Pollefeys, 2005).

In this section, an overview of high quality and GPU stereo matching methods is presented, that has been published in Denker and Umlauf (2011a). Then the most common benchmarks for stereo matching are introduced.

2.3.1 High quality methods

High quality stereo matching methods have been developed based on various techniques. The quality of such methods is compared at Scharstein and Szeliski (2002); Seitz et al. (2006); Strecha et al. (2008). Newer benchmark results are available on the associated websites (Scharstein and Szeliski, 2007; Seitz et al., 2009; Strecha, 2008).

One of the earliest methods in this class is the adaptive least squares correlation of Gruen (1985). In this approach, local affine transformations are estimated using a least squares approximation. Although this method theoretically converges to an optimal solution, the convergence is too slow and the computation too costly due to the size of the linear systems.

The best reconstruction quality is achieved by region growing algorithms, e.g. Furukawa and Ponce (2007); Klaus et al. (2006). These methods are typical for high quality matching algorithms, where a set of good matches is generated using a sparse set of interesting features. Then, these good matches are extended with a growing strategy. The growing operations are iterated in combination with filter operations to control the quality of the matches. Because the growing process is based on an optimization of complex objective functions, these methods do not allow a fast GPU implementation.

An alternative is the phase only correlation of Shibahara et al. (2007). Here, the disparity of matching windows is estimated by the phase difference of the image signal along epipolar lines. The computation of a Fourier transformation is required, which is difficult to implement on the GPU (Moreland and Angel, 2003). This is particularly problematic if the Fourier transform must be evaluated for every pixel of the captured image.

Global optimization of a *Markov random field* (MRF) is used in Campbell et al. (2008). For each pixel multiple depth hypotheses are stored and the best is picked by the MRF optimization. The solution of this NP-hard problem is approximated using a sequential tree re-weighted message passing algorithm (Kolmogorov, 2006). Although the GPU is used to solve several steps of the algorithm, the global optimization makes it much slower than typical GPU methods.

A particle cloud optimization is used by Hornung and Kobbelt (2009) to generate depth representations for each camera image. The particles are aware of depth discontinuous silhouettes and use a special volumetric view space parametrization instead of the usual image-based parametrization of matching windows. Then, these depth representations are combined and rendered in real-time using the GPU.

There are approaches based on dynamic programming, e.g. Lei et al. (2006); Sadeghi et al. (2008). For these methods, fields of matching scores are computed for every epipolar line. Within these fields an optimal path is computed using dynamic programming. The computations of the optimal path can either be done on the CPU or on the GPU, requiring a significant amount of memory.

2.3.2 GPU methods

Much faster methods implement stereo matching algorithms on the GPU using hardware features of the graphics card like mip-mapping.

A typical example for this class of methods is described in Yang and Pollefeys (2005). This approach consists of a set of individual steps of the overall stereo matching process implemented on the GPU. For the matching score, either the sum of squared differences or the sum of absolute differences are used. These matching scores are easily implemented on the GPU, but yield only low quality disparity maps. To exploit the capabilities of the mip-map, a pyramidal matching kernel is used, which does not allow for an independent movement of the individual levels in the pyramid.

A different approach of the same first author is Yang et al. (2002). Here, five calibrated cameras are matched at once. Using the same technique with a reconfigurable array of 48 cameras is described in Zhang and Chen (2004). For this technique, the matching window covers only one pixel to simplify the computations on the GPUs. This local approach is not stable but very fast and avoids all disadvantages of large matching windows.

Another technique for a large number of images is Zach et al. (2006). It is not as fast as the other GPU methods, but includes a volumetric reconstruction of the objects. A plane sweep method is used for depth estimation on non-rectified images.

The method from Cheng and Feng (2005) uses the pyramidal matching kernel and mip-mapping from Yang and Pollefeys (2005) and adds a foreground/background separation on the GPU. This additional step avoids typical artifacts of the pyramidal kernel, like wrong depth estimates for regions with low texture details usually found in the background.

2.3.3 Benchmarks for stereo matching

Benchmarks for stereo matching algorithms allow the comparison of the quality of the resulting depth data. As the applications of stereo matching algorithms differ, there are multiple benchmarks using diverse test datasets. They range from 2 to 363 images, taken from parallel, spherical, or arbitrary camera positions. They also use different result representations, depth maps or meshes, and different comparison methods (Scharstein and Szeliski, 2002; Seitz et al., 2006; Strecha et al., 2008). The following overview of benchmarks for stereo matching algorithms has been published in Denker and Umlauf (2011b).

Middlebury stereo benchmark

One method to compare the quality of stereo matching algorithms is presented in Scharstein and Szeliski (2002). Several pairs of photographs of carefully constructed test scenes are used. All stereo matching algorithms are compared to hand-labeled *ground truth* disparity maps.

The test photographs have a low resolution of 384×288 or 435×383 pixels. Each image pair differs only by a small horizontal displacement of the camera. The viewing directions of both images are exactly parallel. In the used test scenes, most objects have flat surfaces parallel to the image planes of the cameras. This flat structure of the whole scene is necessary to create the ground truth disparity maps. They are build by hand-labeling the contours of all flat regions and computing an alignment for the whole region. In the disparity map, the whole region is marked with the disparity of this alignment. Example images and the corresponding disparity map are shown in Figure 2.10.

Tested stereo matching algorithms have to provide a disparity map. This disparity map is compared with the ground truth disparity map based on the root mean square error. Additionally, the percentage of bad pixels with an error above a

Figure 2.10: The *Tsukuba* dataset from the *Middlebury stereo benchmark* (Scharstein and Szeliski, 2007). The two source images and a reference disparity map.

threshold is computed. Theses scores are computed for the whole image as well as for certain interesting regions. Such regions are texture-less, occluded, and have depth discontinuities.

For algorithms using more than two images, a set of extended datasets was generated using the method described in Scharstein and Szeliski (2003). The extended datasets contain up to nine high resolution photographs (1800×1500 pixels) of constructed test scenes. As for the stereo pairs, the camera is moved horizontally and all viewing directions are parallel. The ground truth is recorded using a structured light approach. A projector casts multiple Gray-code stripe patterns on the scene. The photographs of these patterns are used to compute the exact disparity of each pixel in two reference images.

The official benchmark on the website Scharstein and Szeliski (2007) contains two sets from Scharstein and Szeliski (2002) and two reduced sets from Scharstein and Szeliski (2003). The reduced sets consist of two images and two ground truth disparity images. Their resolution is reduced to 450×375 pixels.

Dense multi view stereo test images

The benchmark approach of Strecha et al. (2008) uses multiple photographs from quite arbitrary camera positions. Outdoor scenes with historic buildings are used as test datasets. Different image sets on the benchmark website contain between 6 and 30 images of the facades of different historic buildings. All pictures are taken using a digital camera without any special positioning device. A laser scanner generated the ground truth geometric data. Two example images and the corresponding depth map are shown in Figure 2.11.

Reconstruction of the internal and external camera parameters is part of the benchmark. Thus, this benchmark can be used without camera parameters, with the internal camera parameters, or with all camera parameters. The ground truth camera parameters are reconstructed from feature markers in the scene using a maximum likelihood estimation.

Figure 2.11: Dataset from *Dense multi view stereo testimages* (Strecha et al., 2008).

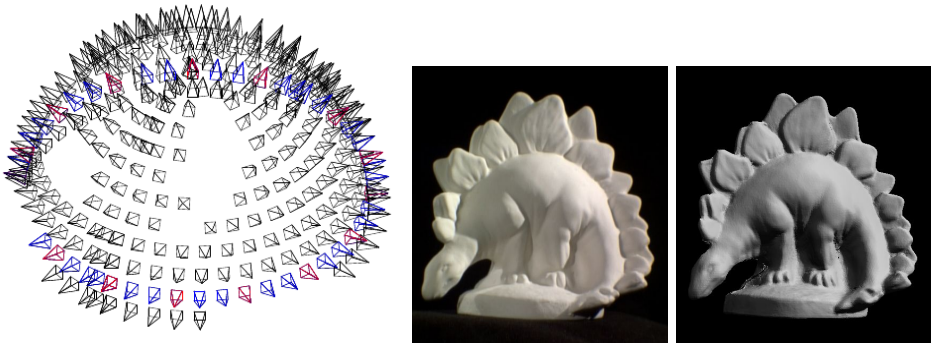


Figure 2.12: Hemisphere of camera positions, photograph, and result mesh from *Middlebury multi view benchmark* (Seitz et al., 2006).

Tested algorithms provide a depth map for each photograph. This depth map is compared to a reference depth map created from the laser scan using the ground truth camera parameters. An error value is computed by dividing each pixel's absolute difference by the local variance of the ground truth laser scan depth map. The benchmark results are presented as a histogram of the different error sizes on the website Strecha (2008).

Middlebury multi view benchmark

The Middlebury multi view stereo benchmark described in Seitz et al. (2006) uses much more data. Pictures from 790 positions on a half-sphere are taken, using the *Stanford spherical gantry* (Levoy, 2002). The spherical gantry uses two computer-controlled arms: one carries a camera, the other a light source. An object is placed in the center between the arms on a turn table. The spherical gantry has a precision of 0.01° and the captured images have a resolution of 640×480 pixels. A *ground truth mesh* G is created from multiple laser scans that are merged using a 3d grid.

Three different datasets of each object can be used for benchmarking. The *full* dataset contains a hemisphere of 317 to 363 images without shadows of the arms, see Figure 2.12. Subsets of these datasets are the *ring* datasets that contain only one ring of 47 to 48 images around the object. The further reduced *sparse ring* datasets contain 16 images on the same ring, equally distributed around the object.

Each algorithm using this benchmark has to provide a *mesh reconstruction* R of the object. The benchmark measures the nearest distance between each point of R to G and of each point of G to R. The latter is only used to evaluate the completeness of R.

CHAPTER 3

Acquisition of 3D Point Data

When real world objects are captured into a virtual model, they have to be scanned. Scanning methods usually capture a set of points, lines, or grids of points. Depth-maps can be interpreted as grids of points because for each pixel a 3d coordinate can be computed from the camera parameters.

This chapter presents the work done for this dissertation in the field of 3d point data acquisition.

3.1 Multi-camera Stereo-matching on the GPU

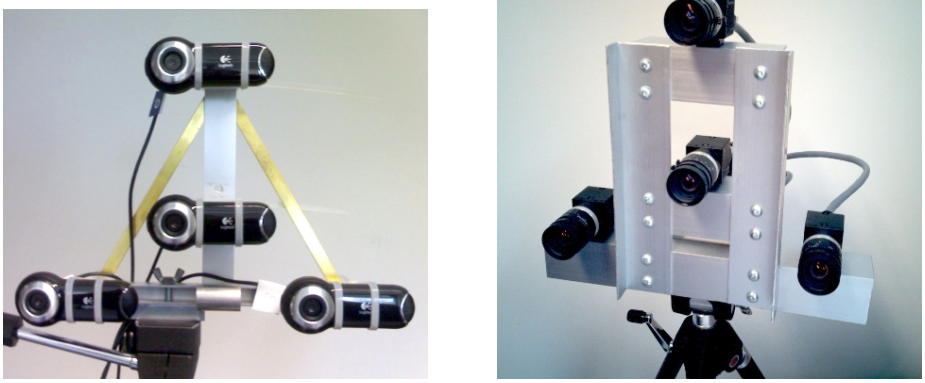
The target application for multi-camera matching on the GPU is 3d face recognition. For face recognition the requirements are somewhere between the high quality methods and the GPU methods described in Section 2.3.2. A trade-off between a high depth map quality and an acceptable speed must be found. The whole reconstruction and recognition needs to be done in less than half a second. A longer delay is not acceptable for the captured person. Nevertheless, the quality of the reconstructed surface needs to be high enough for a reliable recognition of the person.

The work presented in this Section has been published in Denker and Umlauf (2011a). An application of this work in the field of face recognition has been published in Hensler et al. (2011) and is not presented in this dissertation.

3.1.1 The camera systems

A system of four USB Logitech® QuickCam® Pro 9000 cameras was built, see Figure 3.1a. Each camera is used at a resolution of 960×720 at five frames per second. The cameras could yield a much higher resolution, but the bandwidth of the USB 2.0 controllers is limited.

To improve the quality for later face recognition, a second camera system of four Point Grey Flea®2 FireWire 800 cameras was built, see Figure 3.1b. These cameras synchronously capture images at a resolution of 1392×1032 at 15 fps. For



(a) USB camera system. (b) FireWire camera system.

Figure 3.1: For the experiments two systems of four cameras arranged in an upside down Y-constellation are used.

synchronization all four cameras are connected to a single FireWire 800 Bus. Thus, in RGB mode a frame rate of 3.75 fps is possible.

Experiments show that a Y-constellation of four cameras as shown in Figure 3.1 gives the best results. The image of the central camera is used as reference image for matching and texturing. Each possible image pair has a different angle. Otherwise preferred directions of the camera constellation could deteriorate the detection of features along these directions, e.g. an image containing horizontal stripes causes problems for horizontal camera arrangements.

Independently of the used hardware system, this method can be adapted to other camera constellations. This adaption is much easier for camera systems where all cameras are mounted on a plane perpendicular to the viewing direction.

3.1.2 Lens correction

The individual camera images are rectified using a lens correction similar to Devernay and Faugeras (2001). An image is taken from a test pattern containing many straight edges, see Figure 3.2. Contours are reconstructed using a sub-pixel precise edge detection (Devernay, 1995). For each detected contour, a straight line is fitted. The distance of each point on the contour to this straight line is computed and summarized. As a straightness score, the average of the sum of distances for all contours is used.

A radial distortion with a polynomial degree of four is applied to the camera image. The parameters of this distortion are optimized. A simple newton optimization is done for the straightness score, alternating over the parameters of the lens correction. The result set of parameters is used to rectify all images from this camera.

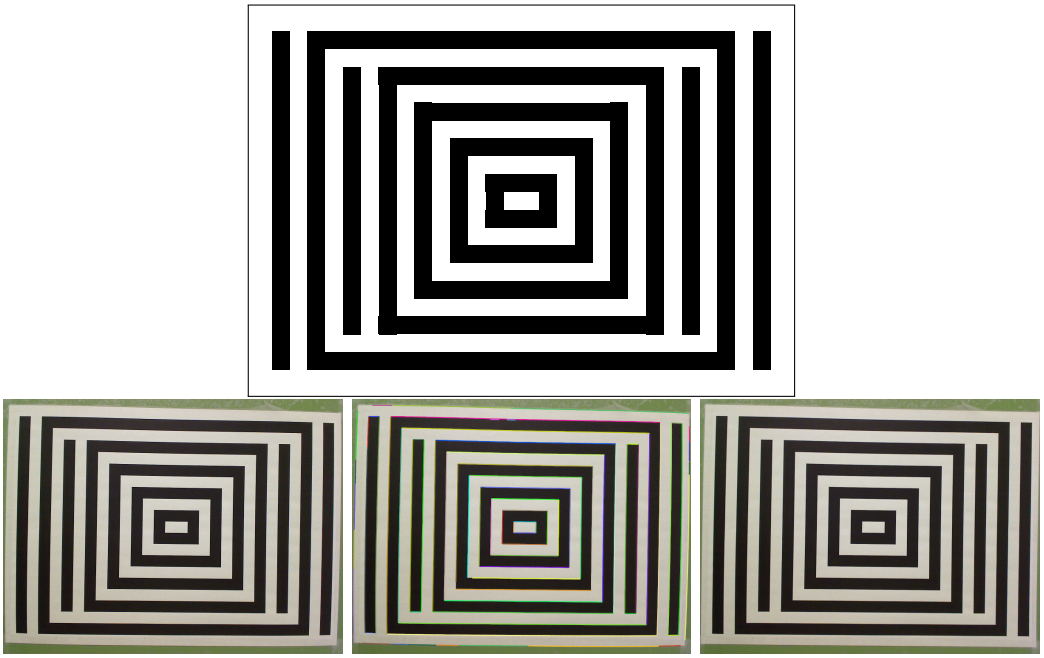


Figure 3.2: The lens correction test pattern (top), uncorrected image (bottom left), detected contours (bottom center), rectified image (bottom right).

3.1.3 Matching

The overall matching process consists of several nested loops shown in Figure 3.3. This process is described in this Section from the inner to the outer loop.

Stereo matching

The aim of stereo matching is to find corresponding points in two images. Usually two square regions, called *matching windows* are compared. These windows are moved over the images to find the best matching position. To identify the best position, a score is computed, that rates the similarity of two matching windows. Similar to Lewis (1995) a *weighted normalized cross-correlation* on RGB color values is used. First the weighted average color $\overline{\boldsymbol{f}}_i$ of the matching window W_i in the i-th image is computed

$$\overline{\boldsymbol{f}}_i = \sum_{(x,y)\in W_i} w(x,y)\boldsymbol{f}(x,y). \tag{3.1}$$

Here $w(x,y) = \cos^2(\pi x/a) \cdot \cos^2(\pi y/a)$ is a weight function that smooths the result to emphasize pixels at the center of the matching window over pixels at the border, and a denotes the matching window size in pixels. Then the weighted auto-

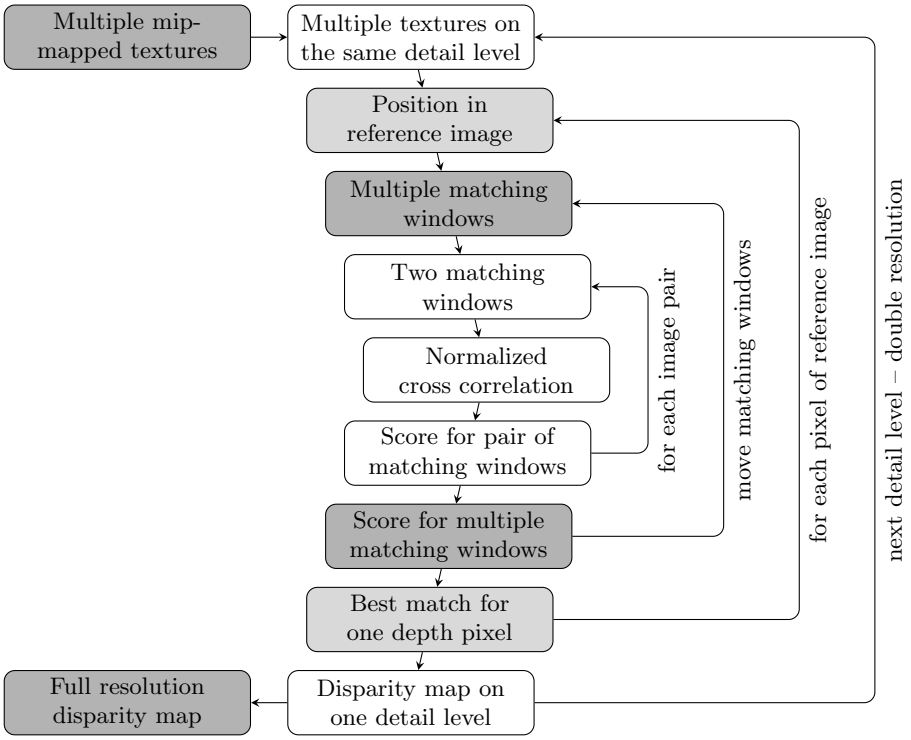


Figure 3.3: Overview of the matching process.

correlation α_i of each matching window with itself is computed as

$$\alpha_i = \sum_{x,y} w(x,y) \left[\boldsymbol{f}_i(x,y) - \overline{\boldsymbol{f}}_i \right]^2 . \tag{3.2}$$

To evaluate the similarity of two matching windows W_i and W_j the weighted cross-correlation $\beta_{i,j}$ is computed

$$\beta_{i,j} = \sum_{x,y} w(x,y) \left[\boldsymbol{f}_i(x,y) - \overline{\boldsymbol{f}}_i \right] \cdot \left[\boldsymbol{f}_j(x,y) - \overline{\boldsymbol{f}}_j \right] . \tag{3.3}$$

The weighted normalized score $\gamma_{i,j}$ is computed as the weighted cross-correlation normalized by the geometric mean of the respective weighted auto-correlations

$$\gamma_{i,j} = \beta_{i,j} / \sqrt{\alpha_i \cdot \alpha_j}. \tag{3.4}$$

Multi-camera matching

Stereo matching evaluates the similarity of two matching windows. This score is extended to a set of n cameras and matching windows by summing up the weighted normalized scores of all possible image pairs. Thus, $n(n-1)/2$ stereo matching

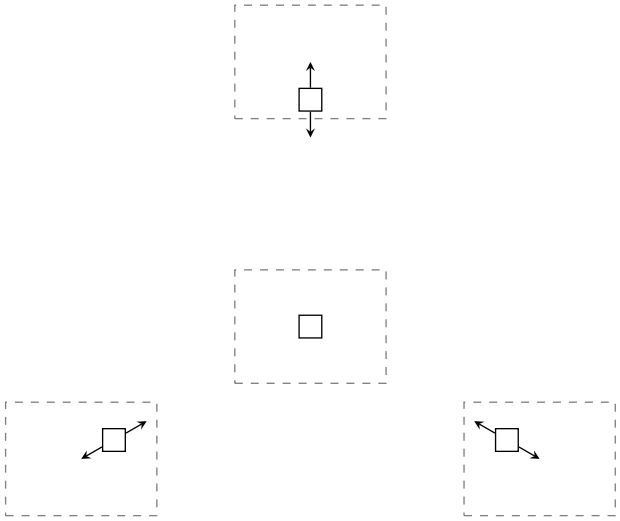


Figure 3.4: Moving the matching windows (solid squares) in all images (dashed rectangles) along epipolar lines (arrows) simultaneously.

operations are required. Camera scores are computed

$$\gamma_i = \sum_{j \neq i} \gamma_{ij} \tag{3.5}$$

and summed up to a total score

$$\gamma = \sum_i \gamma_i - 2 \min_i \gamma_i. \tag{3.6}$$

This eliminates all scores from the worst matching camera to improve robustness to occlusion on one of the cameras. The total score is used to evaluate the similarity of matching windows of multiple cameras simultaneously.

Moving the matching windows

Between the images a disparity estimation is computed to get the depth information. Therefore, the matching windows are moved simultaneously over all images. A total score of each position and the best matching window position with the highest total score are computed. Since the evaluation of all possible positions is too expensive, the movement of the matching window is limited to the epipolar lines projected by the center point of the matching window of the reference image. The image of the central camera is used as reference image, i.e. the matching window on the central image is fixed.

Figure 3.4 shows the simultaneous movement of the matching windows in the other images along the epipolar lines. These movements along the epipolar line

have a step size of one pixel for the used camera configuration. For other camera configurations the step size depends linearly on the distance to the central camera. A set of $3 \leq k \leq 35$ different positions is tested for each matching window. Note that the color values for the score computations are bi-linearly interpolated to allow an exact movement along the epipolar line. The best similarity of the matching windows is marked by the matching window position with the highest score s_{best}. From the position on the epipolar line, the disparity d_{best} of the best match is estimated. The real depth can be computed by reverse projection using the position of the reference camera, the distances to the other cameras, and the disparity.

Sub-pixel matching

To achieve sub-pixel precision for the disparity map a method similar to the sub-pixel accurate edge detection of Devernay (1995) is used. The best disparity is achieved at a local maximum of the total score, i.e. both neighboring scores s_{left} and s_{right} are smaller or at most one of them is equal to s_{best}

$$s_{\text{left}} \leq s_{\text{best}} > s_{\text{right}} \quad \text{or} \quad s_{\text{left}} < s_{\text{best}} \geq s_{\text{right}}. \tag{3.7}$$

Interpolating these three total scores with a quadratic polynomial yields a best sub-pixel score at the global maximum of the quadratic polynomial. This maximum is achieved within half the distance to the neighbor positions. The position of this maximum is the interpolated sub-pixel disparity d_{sub}.

Multi-level matching

The presented stereo matching method generates disparity data for one image at a fixed resolution. To allow large disparities, many possible matching window positions must be evaluated. Because this is computationally expensive, a real multi-level approach is used that can reduce the effort for large disparities. A similar approach in Yang and Pollefeys (2005) uses a matching pyramid. In contrast to the method presented here, the windows on different detail levels cannot be moved independently.

Independent levels allow the re-use of high level information to get a much faster low level disparity computation. The graphics card stores the lens corrected image in a mip-map at eight different resolutions. Each level has half the horizontal and vertical resolution of the one below. All matching windows have a fixed size of 7×7 pixels. A smaller window size increases the noise while a larger size blurs sharp features. Starting on the coarsest resolution level $l = 7$, the disparities of all pixels in the reference images are computed at the same coarse resolution. The matching windows are evaluated at $k = 35$ different positions. Then the image resolution is doubled and the same process starts again, while $k = 1 + 2\lfloor 1.5 + l^2/3 \rfloor$ is reduced quadratically. As starting position for the matching windows on lower levels, the bi-linearly interpolated disparities of the next coarser level are used. Thus, the

matching window moves k pixels around the best position of the previous level.

Deformed matching windows

Square matching windows can only yield good results, if the captured object surface is parallel to the image plane. Every surface not parallel to the image plane generates imprecision. To avoid this the matching windows are deformed to fit the perspective deformation of the object surface. The idea is based on Hattori and Maki (1998), but the multi-level depth information and a projection free computation are used.

The deformation is estimated from the disparity map of the previous multi-level step. First nine disparity values at the corners, the edge midpoints, and the center of the matching window are interpolated. This gives a disparity estimate for every pixel in the actual matching window. Subtracting the disparity at the center of the matching window yields a local displacement for every pixel. This displacement is added to the pixel coordinates before the color values are read. This results in a matching window adapted to the perspective of the previous level without computing any perspective projections. Note, that for planar object surfaces this approach is almost equivalent to the projections used by Hattori and Maki (1998). The difference is that it is based on disparity instead of depth.

Measuring the matching quality

For each resolution level a complete disparity map is computed. So, for each pixel of this map the best total score is stored. Averaging these total scores over multiple resolution levels gives a quality measure for each pixel of the full resolution depth map, see Figure 3.5b. Pixels with low quality measures can be masked for rendering or subsequent computations of the user application.

The quality measure is also used to improve the performance of the multi-level matching. A low quality measure on a coarse matching step usually causes the finer level matches in this region to fail too. Matching calculations are skipped if the quality measure on the next coarser level is too low.

3.1.4 Implementation on the GPU

The method described so far uses images and generates a depth image as result. Therefore, GLSL fragment shaders are used for the GPU implementation. A fragment shader is a program that runs in parallel on the GPU and processes one or multiple texture images into one result image. GPUs which support at least shader model 4.0 are required for the shader operations. The required amount of computations in a single shader run is not feasible on older GPUs.

GPU lens correction

The input data for lens correction are multiple raw camera images. Each raw image is corrected by a shader implementing a lens correction similar to Devernay and

Faugeras (2001), see Section 3.1.2. The resulting corrected images are rendered into separate textures. Each of these textures is then transformed into a mip-map. These mip-maps of the corrected images are used by all subsequent shaders of the presented system.

GPU optimized matching

A single pixel shader run usually computes the color values for one result image, each pixel separately. More complex computations require the combination of multiple shader runs. Three fragment shader programs are used for each step of the multi-level matching.

- The first shader takes the corrected image mip-map and computes the weighted average color of the pixels of a matching window at the actual resolution level. These averages are rendered to separate average textures. This shader is invoked once for every image.
- The second shader takes the corrected image mip-map and the average texture and computes the weighted auto-correlation for the same matching window. Again the result is rendered to a separate auto-correlation texture and the shader is invoked once for every image.
- The third shader takes the average and auto-correlation textures and performs all matching operations, i.e. it moves the deformed matching windows, computes the total score, and finds the best sub-pixel score. The result is rendered as the disparity map, the best total score of the finest resolution and the quality measure to the three color channels of a separate texture. These three shaders are invoked once per resolution level.

Most important strategies used to improve the GPU performance are the pre-calculation of weighted average and weighted auto-correlation just described and the multi-level matching described in Section 3.1.3.

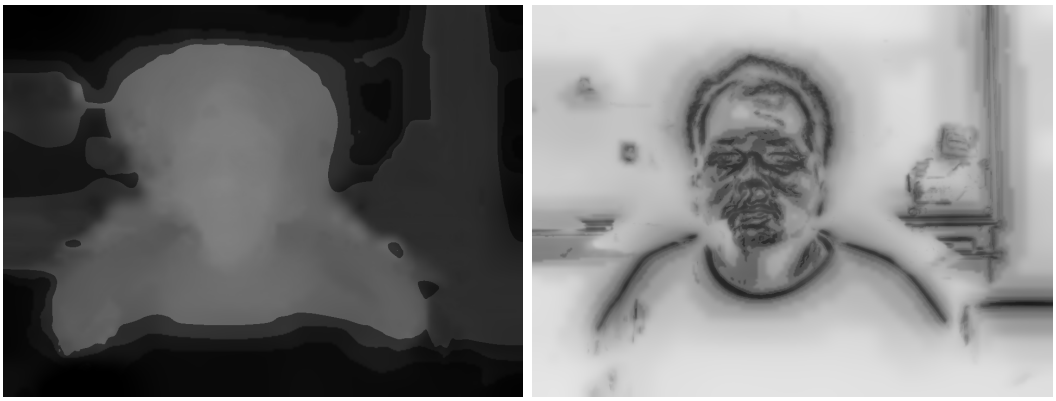
3.1.5 Results

The target application for multi-camera stereo matching is face recognition. In this Section, the results in that area are presented. For easier comparison with other algorithms, the described algorithm is also applied to a well known benchmark for stereo matching.

Face reconstruction

Example images were taken with the USB camera system, see Figure 3.5a. The disparities between these images are very large. The result texture of the fragment shaders holds the disparity map, the best total score of the finest resolution level, and the quality measure, encoded in the color channels, see Figure 3.5b.

(a) The four captured sample images.



(b) Result textures. Disparity map (left) and quality measure (right).

(c) Reconstructed 3d model.

Figure 3.5: Example recorded with the USB camera system.

After transformation of the disparities to depth values, the data can be rendered as a 3d model, see Figure 3.5c. The low quality regions are masked and ignored in this rendering.

A typical problem of stereo matching can be seen at the highlights on the forehead generating small dents, because the reflection is further away from the cameras than the forehead. More diffuse lighting could avoid this problem. The computation for the example images takes an average processing time of 129 ms on an NVidia GeForce GTX 285 GPU. This allows real-time frame rates of 7.5 fps.

A higher resolution of 1392×1032 is achieved by the FireWire camera system. An example image set is shown in Figure 3.6a. Figure 3.6b shows the result textures and Figure 3.6c a 3d model of the resulting depth map. The higher camera resolution yields a better shape quality at the most important regions of the face. Especially the reconstruction of the eye and mouth regions is much more precise.

For this example an average processing time of 263 ms is needed on the same GPU. For images of 30 different persons the average processing time is 254 ms. In most of these images the face region is smaller than in the displayed examples, so the computations are a bit faster. In comparison to the first example, the computation time grows almost linearly with the number of pixels p. This conforms to a runtime of $O(p \log p)$ for the multi-level algorithm: The matching window size, the stretch of the window movement, and the count of image pairs are constant. So the worst case costs for the computations in each depth map pixel is constant. The pixels of the resulting depth map, or smaller versions of it, are computed once for each of the $\log_2(\text{width}) \in O(\log p)$ multi-level steps. Hence the overall count of pixel calculations and the complexity of the algorithm is within $O(p \log p)$.

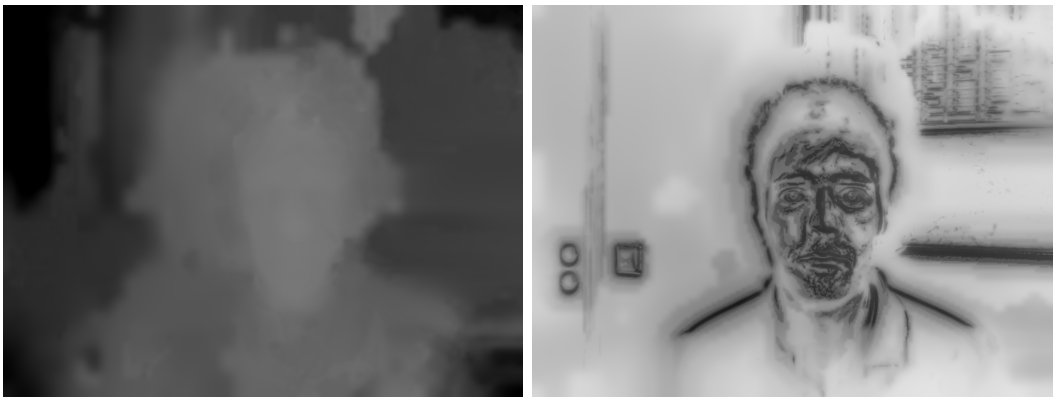
Stereo vision benchmarks

Several benchmarks can be used to compare the quality of stereo matching algorithms, see Section 2.3.3. The presented algorithm is tailored to face reconstruction and contains simplifications that require a planar camera configuration. Thus, it is not comparable to the benchmarks Seitz et al. (2006); Strecha et al. (2008). Furthermore, the algorithm is also tailored to large disparities between the images and achieves a much better reconstruction quality using more than two cameras. So, only a comparison with the results of the *extended datasets* of the Middlebury stereo benchmark (Scharstein and Szeliski, 2007) is relatively fair. However, this benchmark does not provide an official score.

Compared to the algorithms providing results and timings for these benchmark the presented algorithm works much faster. At the same time the quality of the results is comparable to the quality of these algorithms. However, for this comparison some modifications to the algorithm are necessary.

For the Middlebury stereo evaluation (Scharstein and Szeliski, 2007), a modified local version of Multi Hypothesis Matching (Campbell et al., 2008) is integrated to improve the sharpness of edges. The movement range of the matching windows is extended to the depth extrema of the local neighborhood on the last detail level.

(a) The four captured sample images.

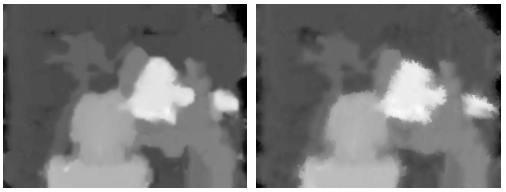


(b) Result textures. Disparity map (left) and quality measure (right).

(c) Reconstructed 3d model.

Figure 3.6: Example recorded with the FireWire camera system.

(a) The extended *Tsukuba* dataset pictures.



(b) Result disparity map of the stereo matching algorithm without (left) and with edge enhancement (right).

(c) Ground truth disparity map (left) and 3d rendering of the result with edge enhancement (right).

Figure 3.7: Results of the extended *Tsukuba* dataset from the Middlebury stereo benchmark (Scharstein and Szeliski, 2007).

Instead of evaluating only the best matching score, the eight best matching scores are stored. A post-processing step re-weights these scores based on the values and depth distances to the best scores in the direct pixel neighborhood. The re-weighting is repeated two times without any global optimization as in Campbell et al. (2008). This multi hypothesis matching is implemented as an post-processing fragment shader on the GPU. The additional shader and the increased range for the matching windows cause a large performance loss. Processing the example images at a resolution of 960×720 pixels takes 900 ms. This is still faster than the other algorithms in Scharstein and Szeliski (2007), but not fast enough for the target application.

Figure 3.7 shows the multi-camera stereo matching applied to the extended *Tsukuba* dataset from Scharstein and Szeliski (2007); Nakamura et al. (1996). The two images in Figure 3.7b show the results from all five input images without and with the additional edge improvement.

3.1.6 Conclusion

The quality of the resulting surface model is sufficient and the processing times are more than sufficient for the target application of 3d face recognition. Additional methods like cross-checking that can be implemented on the GPU could further improve the results.

An application to this stereo matching method is presented in Hensler et al. (2011). There, it is demonstrated that the produced depth data can be used for hybrid face recognition.

3.2 Simulation of Scan Data with Noise

Scanned 3d point data usually has the disadvantage that no ground truth data is available that could be used for error measurements. Therefore, it is necessary to generate synthetic scan data where the exact geometry is known. Simple models of geometric primitives are used to generate synthetic and hand-tracked synthetic scan data.

3.2.1 Synthetic scan data

A simulated line-laser emitter is moved over the surface of the simulated objects. To generate point data, the laser light is intersected with the surface using ray casting. Each of these simulated laser lines consists of 200 points. Thus, 200 rays are created at the emitter and intersected with the simulated object. A realistic amount of 30 lines per second is simulated to allow the assessment of on-line algorithms.

Some of these datasets are shown in Figure 3.8. The simulated plane has an edge length of 20 cm. The cylinder height is 20 cm and the radius is 10 cm; the sphere radius is 10 cm. The added noise is simulated using normal-distributed random numbers. It is scaled according to different levels of standard deviation σ. Two different types of noise are added:

Laser noise: Noise caused by the laser scanner itself is directed along the scanning direction. It is applied to each scanned raw point separately.

Tracking noise: Noise from the tracking of the scanner is added to the scanner position and the raw points in all three spatial directions. It is applied once per scanned line.

3.2.2 Hand-tracked synthetic scan data

The synthetic scan data simulates linear movements of the scanner that cause a rasterized structure of the scan lines. To avoid this effect it is necessary to track hand movements. A magnetic *six degrees of freedom* (6DoF) input controller called *Razer Hydra* (Sixense, 2014), see Figure 3.9, is used to track interactive scans of the simulated objects, see Figure 3.10.

3.3 Multi-camera Laser Scanner

For research, a industrial laser scanner is very expensive. Low cost laser scanners remove the need for expensive measurement equipment. They usually consist of low-cost cameras, line-laser emitters, and card-board sheets with background patterns, see Section 2.1.3. For the previous work in Section 3.1, a synchronized multi-camera system was built. This camera system can be combined with a line-laser emitter to build a low-cost laser scanner without the need for background patterns. The work presented in this Section has been published in Bender et al. (2012).

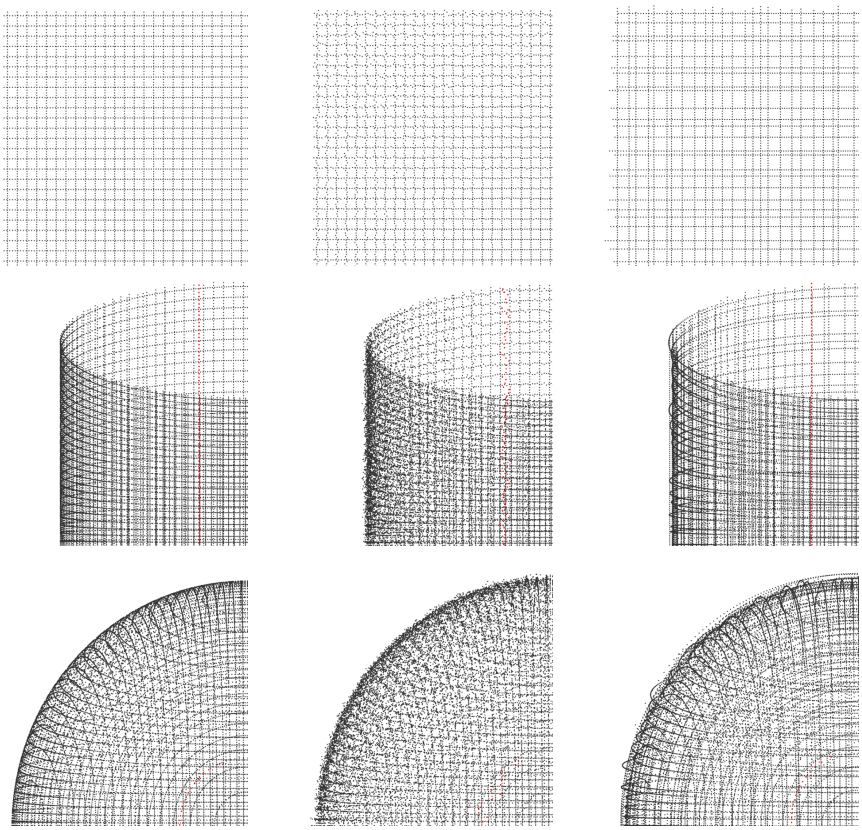


Figure 3.8: Sections of simulated scans without noise (left), with laser noise (middle), and with tracking noise (right).

Figure 3.9: The Razer Hydra (Sixense, 2014), a magnetic 6DoF input controller.

3.3.1 Prerequisites

A triangulation laser scanner usually consists of at least one camera and a line-laser probe. The multi-camera system with four color cameras is used, see Figure 3.11 (left), that was introduced in Section 3.1. This camera system was designed for stereo matching and 3d face reconstruction and recognition. The cameras are mounted in a planar upside down Y-constellation, see Figure 3.11 (right). Thus, each pair of cameras has a different disparity direction to avoid potential problems with features, like a laser line, aligned with a single disparity direction. The four cameras are synchronized such that the cameras take the images at the same time.

To extend this camera system to a laser scanner, one of two line-laser probes is used, see Figure 3.12. Each of these probes consists of a laser emitter and a cylindrical lens. The lens spreads the laser beam to a fan such that a laser line is projected. An additional lens is used to focus the fan to a certain distance. This results in a sharper projection of the laser line. The two probes differ by the color of the laser and the light intensity: The red probe emits a 15mW laser line, the green probe emits a 5mW laser line. Using multiple laser colors allows to adapt to different material properties of scanned objects. The different light intensities are partially compensated by the sensor of the color cameras, which has twice as much green as red pixels.

Unlike usual triangulation laser scanners, in this system the position of the line-laser probe is unknown. The operator holds one of the line-laser probes in his hand and points it towards the scanned object. For each camera, the visible 2d laser line is extracted, see Section 3.3.3. A specialized stereo matching algorithm is used to reconstruct the 3d coordinates of all points on this line.

3.3.2 Calibration

Since a hand-held laser probe is used, there is no calibration of the laser probe required. So, for the calibration of the scanning system the following parameters are needed:

- Camera parameters:
  - Aperture angle α of the cameras.
  - Image height h and width w in pixels of the cameras given by their resolution.
- Relative positions of the cameras.

The aperture angle is computed from the physical width of the area on a planar wall that is visible in the camera image at a one meter distance of the camera to the wall.

The cameras are mounted on a Y-shaped frame of angle plates, see Figure 3.11. Thus, there is one central camera, which will be used as reference for the other three

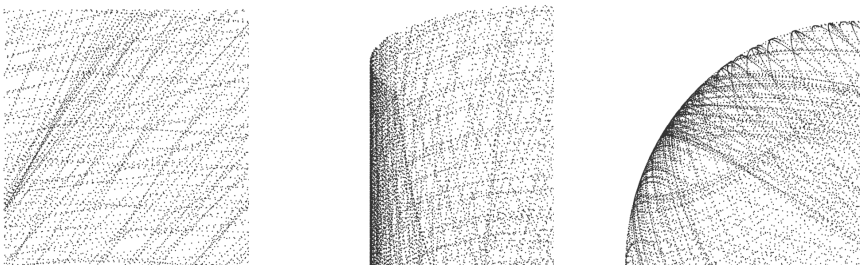


Figure 3.10: Sections of simulated scans scanned interactively with tracking of hand movements.

Figure 3.11: The system of four Point Grey Flea®2 FireWire 800 cameras (left) arranged in an upside down Y-constellation (right).

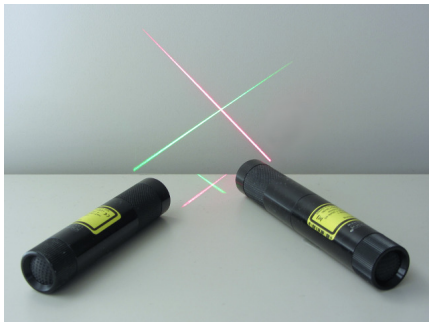


Figure 3.12: Red and green line-laser probe.

so-called outer cameras. In an ideal camera system the cameras are perfectly coplanar, have parallel view directions and the central camera has the same distance $\hat{t}$ to all three outer cameras in physical space. The outer cameras are mounted in (normalized) direction $\hat{t}_i \in \mathbb{R}^2, i = 1, 2, 3$, from the central camera, where the angles of $\hat{t}_i$ to the horizontal image direction are 90°, 210°, and 330°. In image space, the cameras have the relative positions $t_i = t \cdot \hat{t}_i$, where t is the distance measured in pixels of the image centers of the cameras. This can be computed as $t = \hat{t}/s$, where $s = 2\tan(\alpha/2) \cdot z/w$ is the size of one pixel in physical space, if the cameras are placed at a known distance z from a planar wall. This yields the theoretical relative camera positions t_i in image space.

Because of imprecisions in the construction of the Y-frame, the t_i have to be corrected. There is a translational error, because in practice the outer cameras do not have the same distance to the central camera. Due to the construction of the Y-frame, it is assumed that the relative rotational errors of the cameras around the view direction and the horizontal image direction are negligible and the relative rotational errors around the vertical image direction are relatively small. Therefore, the latter can be estimated sufficiently accurate by an additional translational error.

To estimate the translational errors, the true distances in image space of a calibration object captured by all four cameras at the same time are computed. As calibration object a simple red laser-point projected onto a planar wall at the distance z to the cameras is used. This laser-point gives a point p_i, $i = 1, 2, 3, 4$, in each camera's image. In an ideal camera system, p_i, $i = 1, 2, 3$, should appear in the image of the i-th outer camera at position $p_4 + t_i$, if p_4 is the point in the central camera. Thus, the translational error is

$$c_i = p_4 + t_i - p_i, \qquad \text{for } i = 1, 2, 3.$$

To measure p_i for each image, the center of the laser-point is detected as the maximum color value after applying a Gaussian blur filter to the image.

3.3.3 Line extraction

For triangulation scanners based on stereo matching of camera images, the depth of a 3d point can only be computed if it is visible by at least two cameras. Then, stereo matching is the process of finding corresponding points in the camera images of two different cameras at different perspectives. To accelerate this process the matching is limited to points that are on laser lines. So, these laser lines have to be extracted from the images.

For precise depth estimations the extracted lines must be one pixel wide and the points on the extracted lines need to be at sub-pixel accuracy. Furthermore, the extraction process must be robust to noise and must execute in real time. To satisfy these requirements, techniques from Clode et al. (2004); Devernay (1995) are adopted in Steps 1.2 and 1.3 of the line extraction algorithm below.

The line extraction algorithm is applied to every image and consists of five steps

described below. It takes as input an image, which is given by $I : \{-w/2, \ldots, w/2\} \times \{-h/2, \ldots, h/2\} \to \mathbb{R}^3$, $(x, y) \mapsto (r, g, b)$, where (x, y) are pixel coordinates. The three coordinate functions I_r, I_g, and I_b of I represent the three color channels of the image.

Line extraction algorithm

1.1 Binarize the source image's red channel I_r (analogously for the green channel)

$$B_I(x, y) = \begin{cases} 1, & \text{if } I_r(x, y) > t_I \\ 0, & \text{otherwise.} \end{cases}$$

The threshold t_I is set manually. For the experiments, $t_I = 0.165$ is used.

This step removes most of the information from the image, that is not necessary for the line extraction. In B_I the laser lines are several pixels wide.

1.2 Convolve the binary image B_I with the phase coded disc O_{PCD}

$$Q(x, y) = \frac{1}{\pi r^2} \sum_{u=-r}^{r} \sum_{v=-r}^{r} B_I(x + u, y + v) \cdot O_{\text{PCD}}(u, v) \in \mathbb{C}, \tag{3.8}$$

where O_{PCD} is defined in Equation (3.9) and the radius r of the disc is chosen to be larger than the maximum width of the laser line.

This step yields an image with maximal values at the line centers. This line of maxima is exactly one pixel wide.

1.3 A sub-pixel accurate non-maximum suppression NMS is applied to $|Q(x, y)|$ along direction $\text{Arg}(Q(x, y))/2$ to mask the line of maxima from the rest of the image

$$N(x, y) = \text{NMS}\left(|Q(x, y)|, \frac{1}{2}\text{Arg}(Q(x, y))\right) \in \mathbb{R},$$

where $\text{Arg}(z)$ is the principal value of the phase of a complex number z.

1.4 Binarize $N(x, y)$

$$B_N(x, y) = \begin{cases} 1, & \text{if } N(x, y) > t_N \\ 0, & \text{otherwise.} \end{cases}$$

The threshold t_N is set manually. For the examples, $t_N = 0.15$ is used.

This step yields a binary image with sequences of line center points, that are one pixel wide and have well defined start- and end-points.

1.5 Subsume the center points in B_N to line segments. Line segments that are shorter than 50 points are ignored.

This line extraction algorithm is implemented in C++ and OpenGL Shading Language GLSL. The GPU is used for

- binarization of the images (Steps 1.1 and 1.4),
- convolution with the phase coded disc (Step 1.2), and
- non-maxima suppression (Step 1.3).

Only the line tracing in Step 1.5 is computed on the CPU.

Phase coded disc

In Step 1.2 a convolution with a phase coded disc is used (Clode et al., 2004), which is defined as

$$O_{\mathrm{PCD}}(u,v) = \begin{cases} e^{2i\,\mathrm{Arg}(u+iv)} & \text{, if } \sqrt{u^2+v^2} \leq r, \\ 0 & \text{, if } \sqrt{u^2+v^2} > r, \end{cases} \tag{3.9}$$

where $e^{2i\,\mathrm{Arg}(u+iv)}$ is the exponential representation of a complex number whose phase is twice the phase of $u+iv \sim (u,v)$. Note that $\mathrm{Arg}(u+iv)$ is computed by $\mathrm{atan2}(v,u)$, the four-quadrant arctangent function.

Due to the doubling of the phase in Equation (3.9), the phase angles rotate twice through $[0, 2\pi]$ as (u,v) rotates once around the origin on the phase coded disc O_{PCD}, see Figure 3.13. This has the effect that points, whose phase differs by 90°, have opposite phases (180° difference) on O_{PCD}. Thus, they attenuate in the convolution from Equation (3.8). On the other hand, points with the same or opposite phases have the same phase on O_{PCD}. Thus, they amplify in the convolution from Equation (3.8).

For the convolution of O_{PCD} with the binarized image B_I this has the effect, that the magnitude $|Q(x,y)|$ is relatively large, if at (x,y) the laser line contains the center of O_{PCD}, see Figure 3.13 (left). Otherwise, $|Q(x,y)|$ is relatively small, see Figure 3.13 (right). So, after convolving B_I with O_{PCD}, the maxima of $|Q|$ are located on the center of the laser line. However, lines with 90° corners cannot be detected with this approach.

Non-maxima suppression

To find sub-pixel accurate maxima in the convolution images the approach of (Devernay, 1995) is adapted in two ways. First, the magnitude of Q is used instead of the gradient's magnitude. Second, the normal $L^{\perp}$ to the laser line direction L is used instead of the gradient direction. The direction of the laser line $L(x,y)$ is determined by half the phase angle of $Q(x,y)$

$$L(x,y) = \frac{1}{2}\mathrm{Arg}(Q(x,y)).$$

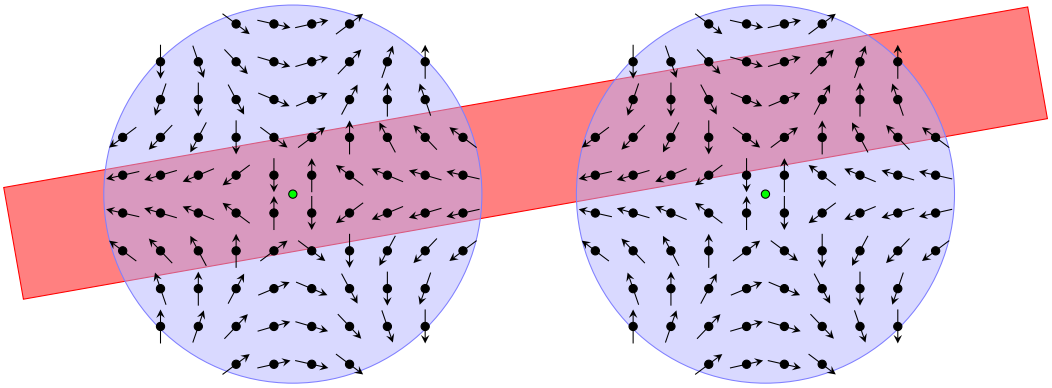


Figure 3.13: Phase coded discs with red laser line and small arrows visualizing the complex phase angles.

Thus, $L^{\perp}(x, y)$ is perpendicular to L at (x, y).

Non-maxima suppression algorithm

2.1 Denote by B the intersection of the line through a pixel A in direction $L^{\perp}$ with a line segment through the neighbor pixels A_i and A_{i+1} for $i \in \{1, \ldots, 4\}$, as displayed in Figure 3.14. Analogously, C is the intersection in the opposite direction from A. Thus, the values of $|Q|$ at B and C are linearly interpolated between the values at A_i and A_{i+1} respectively A_{i+4} and A_{i+5}.

2.2 If there is no maximum at A compared to B and C, A is ignored in the result.

2.3 If there is a maximum at A, the sub-pixel accurate position is computed as the position along line $L^{\perp}$ of the maximum of the quadratic interpolant of the values at A, B, and C, see Figure 3.15.

Thus, the decimal places of the sub-pixel coordinates of the maximal value and the maximal value of $|Q|$ are stored per pixel.

Line tracing

The extracted line center points in B_N have to be assigned to line segments. Thus, the output of Step 1.5 is a list of line segments $(s_k)_k$ for each image. Each line segment s_k is a list of line center points $(p_{k,l})_{l=0}^{n_k}$.

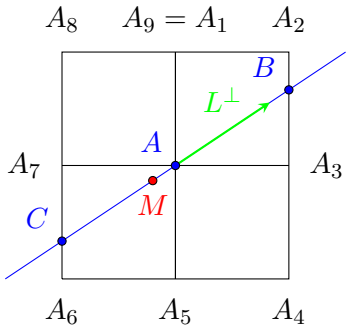


Figure 3.14: Sub-pixel accurate position of the maximum M on line segment CB ((Devernay, 1995)).

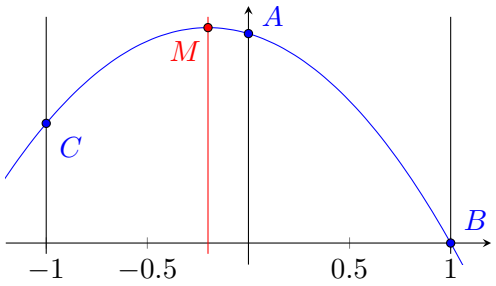


Figure 3.15: The quadratic interpolation of the values at A, B, C. At M is the maximum of the interpolant.

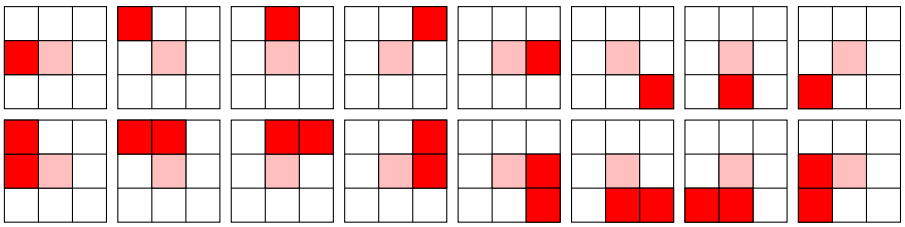


Figure 3.16: Start point patterns of 3×3 pixels: The white pixels have value 0 in B_N, the other pixels have value 1. Thus, the reddish pixels belong to a laser line. The pink pixel is the query pixel.

Line tracing algorithm

3.1 Identify start points $p_{i,0}$ in B_N by matching with the 3×3 pixel patterns in Figure 3.16 and generate a new segment s_k containing the start point $s_k = (p_{k,0})_l$. Start points, which are already part of a line segment, are ignored.

3.2 For a segment $s_k = (p_{k,0}, \ldots, p_{k,l})$ check the 8-neighborhood of point $p_{k,l}$ in B_N for pixels with value 1. If direct and diagonal neighbors are found, the former are preferred. If a new point $p_{k,l+1}$ is found, it is appended to the corresponding line segment $s_k = (p_{k,0}, \ldots, p_{k,l+1})$. Visited points are tagged to avoid multiple visits in Steps 3.1 and 3.2.

3.3 Repeat Step 3.2 until a point is identified as end-point matching the 3×3 pixel patterns in Figure 3.16.

3.4 An extracted line segment s_k is ignored, if it contains less than 50 points.

3.3.4 Depth estimation

Similar to the stereo matching method in Section 3.1, the spatial depth of the point data is reconstructed from the disparities between images. This requires a calibrated camera system whose lenses are corrected. For the lens correction the method from Devernay and Faugeras (2001) is used to correct all camera images to a central projection in a pre-processing step, see Section 3.1.2. Then, the calibration as described in Section 3.3.2 is applied.

At every instant of time the camera system generates a quadruple of images containing one image I^i, $i = 1, 2, 3, 4$ from each camera. Accordingly, the above line extraction algorithm yields a quadruple of e.g. B_N^i where the super-script indicates the i-th outer camera for $i = 1, 2, 3$ or the central camera for $i = 4$. The line extraction step (Section 3.3.3) provides three representations of the laser line per camera image: the line segments s_k^i, the binary image $B_N^i(x, y)$, and the sub-pixel accurate laser line center point in $N^i(x, y)$. For each point $p_{k,l}^4$ of the line segment s_k^4 of the central camera image, a depth is estimated using the images of the outer cameras:

4.1 Corresponding points in two camera images from two different cameras are identified along epipolar lines.

4.2 For each pair of corresponding points the depth is estimated by an inverse projection.

Epipolar lines

After the calibration it is assumed that all four cameras of the camera system are co-planar and have parallel view directions. Therefore, a point $\hat{X}$ at infinite depth in physical space will have the same image coordinates X_i, $i = 1, 2, 3, 4$, in all four

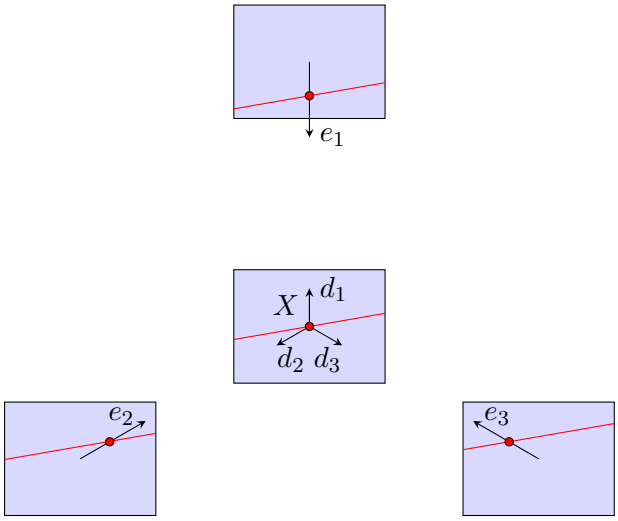
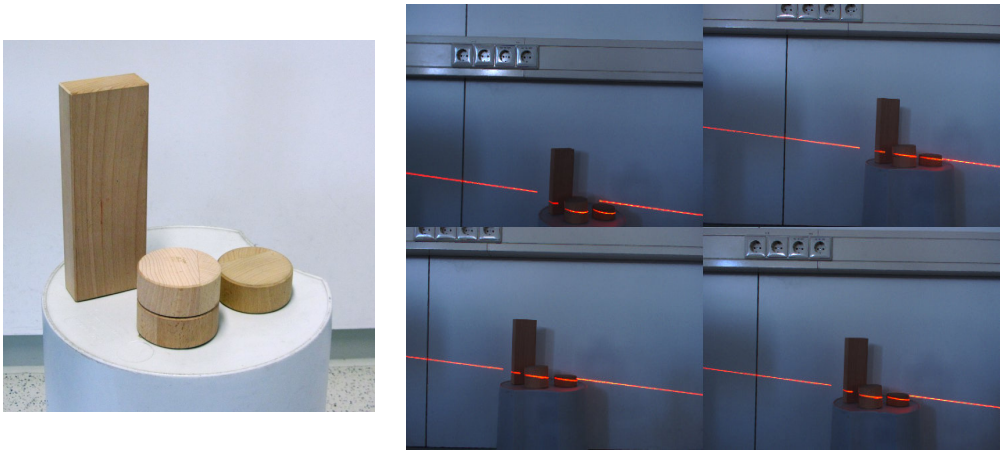


Figure 3.17: Schematic illustration of the four camera images with projections of epipolar lines e_i of a point X (red point) in the central camera's image on a laser line (red line), see Section 3.1.

camera images I^i. A point $\hat{Y}$ not at infinity with image coordinates $Y_4 = X_4$ has image coordinates $Y_i \neq Y_4$, $i = 1, 2, 3$, in the outer cameras. As $\hat{Y}$ moves closer, Y_i, $i = 1, 2, 3$, moves along the negative camera direction $-d_i$. Thus, $\hat{Y}$ traces out a ray in physical space pointing away from the central camera. This line is called the *epipolar line* of X_4. The central projection of an epipolar line in the outer camera images is a straight line, too. Figure 3.17 shows the projections $e_i(X)$ of an epipolar line of X for the camera system, schematically.

For every point $p^4_{k,l}$ the epipolar line is computed and projected to the outer camera images. The resulting epipolar line projections $e_i(p^4_{k,l})$, $i = 1, 2, 3$, are rendered to image I^i using the Bresenham algorithm (Bresenham, 1965). If during this rendering a pixel is set that is also set in $B^i_N(x, y)$ an intersection of the epipolar line e_i with a line segment in I^i is detected. The pixel distance between $p^4_{k,l}$ and the sub-pixel accurate laser line position from $N^i(x, y)$ at the intersection is the so-called *disparity* $d_i(p^4_{k,l})$. So, there are up to three disparities d_i, $i = 1, 2, 3$, for each $p^4_{k,l}$.

To improve the quality of the reconstruction, subsequently the average $d(p^4_{k,l})$ of the two disparities that are closest to each other is used. If there are less than two disparities or the two closest disparities differ too much, the computations for $p^4_{k,l}$ are aborted.



(a) (b)

Figure 3.18: Test scene consisting of four wooden bricks (a) and the four images of the test scene scanned with a red laser line captured by the camera system (b).

Inverse projection

With the average disparity d, the depth c_z in physical space of $p^4_{k,l}$ is estimated by

$$c_z = \frac{w\hat{t}}{2\tan\left(\frac{\alpha}{2}\right)d},$$

where α is the aperture angle of the camera, w the image width in pixels, and $\hat{t}$ the camera distance in physical space. The depth c_z together with the pixel coordinates $p^4_{k,l}$ allow an inverse projection of $p^4_{k,l}$ to 3d coordinates $[c_x, c_y, c_z]^T$ in physical space as

$$\begin{bmatrix} c_x \\ c_y \end{bmatrix} = 2\,\frac{c_z \cdot p^4_{k,l}}{w}\tan\left(\frac{\alpha}{2}\right) = p^4_{k,l}\frac{\hat{t}}{d}.$$

3.3.5 Results

To demonstrate the effectiveness of the laser scanner system, a test scene of three wooden bricks shown in Figure 3.18a was scanned. The exposure of the cameras was reduced to achieve a better contrast between the laser line and the surroundings, see Figure 3.18b. During a period of five minutes, ca. 210,000 points were captured at a rate of two image-quadruples per second.

Figure 3.19 shows a front and a top view of the computed point cloud. The overall quality of the data allows to recognize the shapes of the different wooden bricks, especially the one in the foreground. The reconstruction has a higher quality

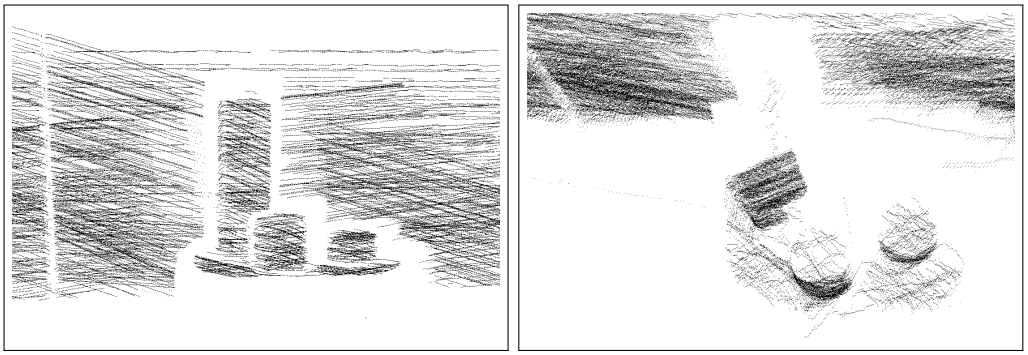


Figure 3.19: Point cloud from two different perspectives (left: front view, right: top view)of the test scene in Figure 3.18 scanned with a red laser line.

and is more robust than the pure stereo matching method in Section 3.1.

The scan lines in the point clouds show some small periodic noise in view direction of the cameras. This might be caused by three effects.

Aliasing at Step 1.2 of the line extraction, Section 3.3.3: The binarization $B_I(x, y)$ in Step 1.1 creates an aliased image of the laser line. In particular, if the aliasing occurs on both sides of the line simultaneously, this may cause small steps in the detected line centers.

Intersection of e_i with the line segments in I^i in Section 3.3.4: The laser line position from $N^i(x, y)$ is at sub-pixel accuracy while the projected epipolar line e_i is not. Although this approximation does not affect the disparity d_i very much, a more precise intersection of the line segment with e_i could improve the results.

Calibration in Section 3.3.2: The laser point is not detected at sub-pixel accuracy.

Another effect, that can be observed in the scan data, is that some of the scanned laser lines appear more than once in the data at slightly different depths. This is caused by blurry camera images, e.g. by motion blur. Additional filtering after the detection of the line centers in Step 1.2 of the line extraction algorithm in Section 3.3.3 could be used to avoid these artifacts.

3.3.6 Conclusion

In this Section it was demonstrated how to built a laser scanner device at low costs using an existing camera system. The presented reconstruction algorithm runs on the GPU and is fast enough to compute 3d point data during the scan. Despite the remaining problems (see Section 3.3.5) the reconstruction yields better quality and is more robust than the pure stereo matching method in Section 3.1.

The laser scanner is limited to scanning from a single camera position. This only allows to capture one side of an object. For complete scans of objects it is necessary to either move the camera system or rotate the object e.g. on a turn table. Combining such scans from different directions usually requires an *iterative closest point* (ICP) algorithm.

It is possible to improve the quality of the scanned results in a post-processing step. All points on a laser line lie in the plane of the laser line fan. Thus, fitting this plane to each scan line and projecting the points onto this plane along the view direction of the cameras will probably solve most problems described in Section 3.3.5.

CHAPTER 4

On-line Reconstruction

A hand-held laser scanner captures a stream of 3d point data, see Sections 2.1 and 3.3. On-line triangulation (Denker et al., 2008, 2011) provides a mesh representation of this data. To allow editing, the data representation of most CAD application consists of *constructive solid geometry* (CSG) and free form surfaces. CSG geometry is based on simple geometric primitives, that are hierarchically combined to more complex objects by boolean operations. The on-line reconstruction algorithms in this chapter aim to reconstruct a set of such simple geometric primitives and their parameters on-line from a stream of 3d point data.

One of the first approaches of reconstructing CAD data from triangulation laser scanners is presented in Agin and Binford (1976), where cylinders are fitted to point data from a laser scanner. To reconstruct more complex objects a segmentation of the surface is necessary. An extensive overview of surface segmentation methods is provided in Shamir (2008).

The underlying data structures are either meshes, e.g. Wu and Kobbelt (2005), or a k-nearest neighbor graph, e.g. Yamazaki et al. (2010), for an efficient local surface analysis. For CAD applications usually a segmentation by surface type is computed. The Gaussian sphere is used in Benkő and Várady (2002) to segment surfaces by their dimensionality, whereas variational surfaces are used in Wu and Kobbelt (2005).

Early methods for the reconstruction of CAD geometry are discussed in Várady (1997). They are based on the reconstruction of polygonal boundary models and the fitting of simple surfaces and free form geometry. The method presented in Benkő et al. (2001) focuses on the reconstruction of rotational and translational surfaces and blends. In Wu and Kobbelt (2005) variational surfaces are used to reconstruct implicit representations of geometric primitives. All these afore mentioned methods do not work for on-line computations of point streams.

Iterative methods for geometry reconstruction often apply stochastic algorithms like Random Sample Consensus (RANSAC) fitting. They are used for the reconstruction of geometric primitives (Schnabel et al., 2007) or super-quadrics (Biegelbauer and Vincze, 2007). These RANSAC based methods work on random sub-sets

of the data. Iteration between segmentation and reconstruction is used in Vančo et al. (2008). Here, based on quadric fitting the segmentation of an unorganized point cloud is iteratively improved. Complete data sets are required for each iteration.

Stream processing of point data is discussed in Pajarola (2005); Boesch and Pajarola (2009). A sweep line approach sequentially processes large point sets with a set of geometric operators. Thus, the point sets have to be pre-sorted along one spatial direction.

Unorganized streams of point data are generated by hand-held laser scanners. Such streams can be triangulated on-line (Bodenmüller and Hirzinger, 2004). The data is reduced to a set of vertices that are almost uniformly distributed. Then, a surface mesh is generated on-line connecting these vertices. This approach was extended by Denker et al. (2008, 2011) with a multi-level data structure. Thus, the mesh can be adapted to heterogeneous point densities in the data.

4.1 Data Structures and Parallelized Implementation

The basic data structure for the on-line reconstruction is a *ball tree* as described in Denker et al. (2008, 2011). This data structure stores and processes the data stream from the laser scanner and prepares it for all subsequent reconstruction steps.

This Section provides a short overview of the ball tree data structure as described in greater detail in Denker et al. (2011). Afterwards, some improvements to this original data structure are described.

4.1.1 Ball tree generation

The data stream from the laser scanner consists of 3d raw points and orientation data of the laser probe. The raw points are assigned to so-called *neighborhood balls* or *n-balls* β which are defined as

$$\beta(\boldsymbol{c}, r) = \{\boldsymbol{x} \in \mathbb{R}^3 : \|\boldsymbol{x} - \boldsymbol{c}\| < r\}$$

with center $\boldsymbol{c}$ and radius r. These n-balls might intersect. As the raw points are streamed in, each raw point $\boldsymbol{p}$ is associated to the n-ball $\beta(\boldsymbol{c}, r)$ that contains $\boldsymbol{p}$ with minimal distance $\|\boldsymbol{p} - \boldsymbol{c}\|$. If no such ball exists a new n-ball $\beta(\boldsymbol{p}, r)$ is generated with center $\boldsymbol{p}$ and maximal radius r such that $\beta(\boldsymbol{p}, r)$ does not contain any other n-ball center.

N-balls are organized in an octree data structure, the so-called *ball tree*. An n-ball is associated to the octree region that contains its center. The edge length of this region equals the radius of its associated n-balls. So, all n-ball radii are dyadic fractions of the edge length e_O of the initial scan area, i.e., $r = e_O/2^n$ for $n \in \mathbb{N}$.

Heuristics are used to balance the data structure (Denker et al., 2011). The heuristics aim at a nice coverage of surfaces with neighborhood balls. They also prevent that too many data points are associated with the same n-ball.

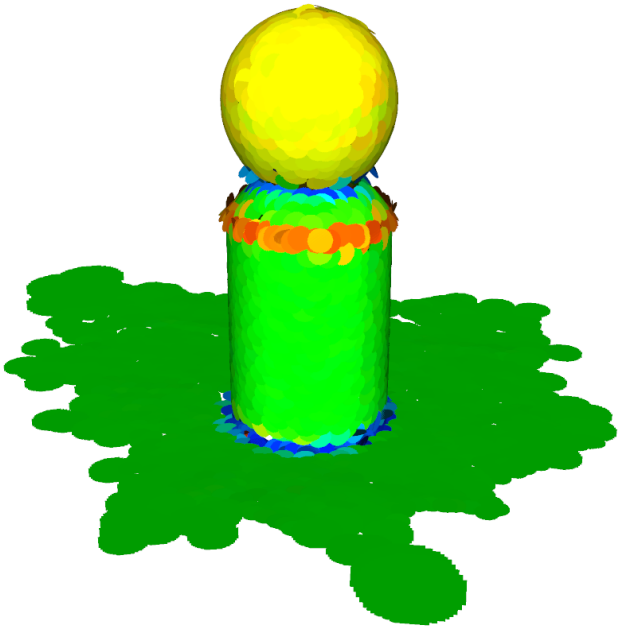


Figure 4.1: Splat based visualization (Rusinkiewicz and Levoy, 2000) of the geometry of n-balls for a scan of a wooden toy. The colors indicate the Gaussian curvature $K = \kappa_1\kappa_2$.

4.1.2 Local geometry estimation

For each n-ball β estimates for the local normal and principal curvatures are computed. The normal $\boldsymbol{n}_\beta$ is computed by principal component analysis (PCA) of the raw points of a neighborhood of n-balls in the ball tree. To compute the principal curvatures κ_1 and κ_2 and principal curvature directions $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ a quadratic polynomial P_β is fitted to these raw points based on the local tangent plane defined by $\boldsymbol{n}_\beta$. The Weingarten map of P_β yields κ_1, κ_2, $\boldsymbol{u}_1$, and $\boldsymbol{u}_2$. Note, that these estimates are only used if the ratio of the smaller eigenvalues of the PCA is less than $1/2$. Otherwise it is assumed that the local point data does not represent a reconstructable surface.

In Denker et al. (2011) every n-ball β corresponds to one vertex $\boldsymbol{v}_\beta$ of a triangle mesh. Its position is computed as the arithmetic mean of raw points of β projected to P_β. However, for the on-line reconstruction no triangle mesh is constructed. Still, $\boldsymbol{v}_\beta$ is used to represent the geometry of β in subsequent reconstruction steps. Similar to Yamazaki et al. (2010), each n-ball holds a list of 20 nearest vertices in the ball tree.

A visualization of n-balls and their local geometry is shown in Figure 4.1. A scan of a wooden toy is displayed using a splat visualization (Rusinkiewicz and Levoy, 2000). The splats for an n-ball $\beta(\boldsymbol{p}, r)$ are defined by $\boldsymbol{v}_\beta$, $\boldsymbol{n}_\beta$, and r. The color is computed from κ_1 and κ_2.

4.1.3 Segment data structure

A *segment* is a set of n-balls of approximately the same geometric type. The implementation is based on a hash table (Qt Project Hosting, 2013). Thus, look-up and insertion operations have amortized computational complexity $O(1)$.

Each segment maintains a set of its neighbor segments. This set contains all segments containing n-balls that are neighbors to n-balls in the segment. Neighbor segments are added to this set when n-balls are added or modified. At the same time a back link is added to the neighbor segment's set. Neighbor entries are only removed upon segment deletion.

For the association of n-balls to segments also a hash table is used. Furthermore, a hash table containing each n-ball is used to maintain all geometric quantities that are associated to this n-ball.

4.1.4 Parallel implementation

In contrast to Denker et al. (2011) the implementation used here is more parallel to achieve the necessary speed for real-time on-line reconstruction. The different parts of the software run independent in individual threads. To minimize locking, all data is exchanged using operation queues. The queued operations are *add*, *delete*, and *update*. Each thread holds its own reduced copy of the geometric data.

Each time a raw point is added to an n-ball, its local geometry is updated. The updates of n-balls are independent and can be done in parallel using multiple threads. Because these changes are often small, the respective update might be postponed. Thus, a priority queue is used to process the control the sequence of updates. Each n-ball β has a priority p_β assessing the importance of an update. Adding a raw point increases p_β by one, if the ball has a stable normal, or by two otherwise. More important operations like adding or deleting a neighbor increase p_β by five. Whenever an update is completed, p_β is reset to zero.

A global priority limit $p_{\max}$ is used to decide which n-balls are updated. $p_{\max}$ is adjusted depending on how much updates are currently processed. This strategy assigns the available processing time to important tasks without wasting to much of it for minor updates.

4.2 Segmentation Using Local Geometric Values

The ball tree data structure provides a local geometry estimation, see Section 4.1.2. For the segmentation of the scan data the n-balls are classified by their local geometry. Based on this classification, an on-line region growing algorithm is applied to obtain segments of the same surface type. Consistency criteria within the segments are used to improve the segmentation.

The segmentation algorithm presented in this section has been published in Denker et al. (2013).

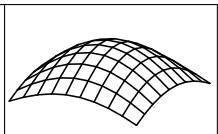
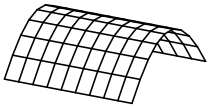
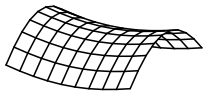
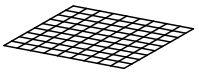
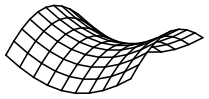
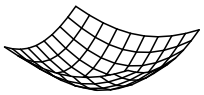
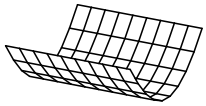
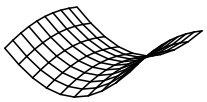
	$K > \varepsilon_K$	$\lvert K\rvert \leq \varepsilon_K$	$K < -\varepsilon_K$
$H < -\varepsilon_H$	 peak	 ridge	 saddle ridge
$\lvert H\rvert \leq \varepsilon_H$	—	 plane	 minimal surface
$H > \varepsilon_H$	 pit	 valley	 saddle valley

Table 4.1: Eight surface types determined by the Gaussian and mean curvature (Besl and Jain, 1988).

4.2.1 N-ball classification

In Besl and Jain (1988) a classification of a surface based on the Gaussian and mean curvature is introduced. The resulting eight surface types are shown in Table 4.1. This classification is applied to each n-ball. With the principal curvatures κ_1 and κ_2 from Section 4.1.2 the required Gaussian $K = \kappa_1\kappa_2$ and mean curvature $H = (\kappa_1 + \kappa_2)/2$ are computed.

4.2.2 Segmentation criteria

N-balls of the same surface type are clustered to surface segments. Depending on the surface type, different segmentation criteria are used in this clustering step. These different criteria are explained in the following.

The ratio of the two smallest eigenvalues resulting from PCA allows the assessment of the quality of estimated normals during the estimation process itself. Whenever a certain quality threshold ε_q is not satisfied, the n-ball is not used for segmentation.

Criterion for a planar segment

To cluster n-balls to a planar segment the n-ball normals $\boldsymbol{n}_\beta$ are used. Two n-balls are clustered to the same planar segment, if the angle between their normals is less than a threshold ε_n.

Two planar segments with the same normal do not necessarily belong to the same plane. If such planar segments are close to each other, the normal angle will not be sufficient to discriminate these segments. If β_1 and β_2 are two n-balls with parallel

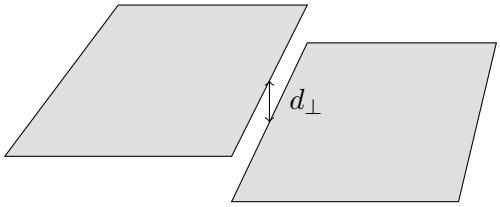


Figure 4.2: Perpendicular distance $d_\perp$ between two neighboring plane segments.

normals, the perpendicular distance

$$d_\perp(\beta_1, \beta_2) = (\boldsymbol{n}_{\beta_1} + \boldsymbol{n}_{\beta_2}) \cdot (\boldsymbol{v}_{\beta_1} - \boldsymbol{v}_{\beta_2}) / \|\boldsymbol{n}_{\beta_1} + \boldsymbol{n}_{\beta_2}\|$$

is computed. The n-balls β_1 and β_2 are clustered to the same segment, if $d_\perp(\beta_1, \beta_2)$ is less than the threshold $\varepsilon_{d_\perp}$. Figure 4.2 illustrates the perpendicular distance $d_\perp$ of two neighboring planes.

Principal curvature criterion

For spherical and cylindrical segments the principal curvature is used as a criterion for segmentation. An ideal one of these segments would have the same values of κ_1 and κ_2 for each n-ball.

Spheres
All points on a sphere have the same maximum and minimum curvatures. Due to measurement noise of the laser scanner, a threshold ε_s for the curvature of n-balls on spheres is used. To enhance the robustness of this criterion, the average maximum $\bar{\kappa_1}$ and minimum curvature $\bar{\kappa_2}$ of all n-balls belonging to one segment is compared to the principal curvature κ_1 and κ_2 of a candidate n-ball

$$|\kappa_i - \bar{\kappa}_i| < \varepsilon_s \quad \text{for } i = 1, 2.$$

Cylinders
All points on a cylinder have the same maximum principal curvature κ_1. The minimum principal cuvature κ_2 is 0 for all cylinders. Again, a candidate n-ball's maximum principal curvature κ_1 is compared to the segment's average maximum curvature $\bar{\kappa}_1$

$$|\kappa_1 - \bar{\kappa}_1| < \varepsilon_c.$$

On-line computation of average curvatures

N-balls might be added and deleted from segments in an arbitrary order. Simultaneously the average principal curvature of segments must be updated. If $\bar{\kappa}_i^n$ denotes the average principal curvature of a segment s with n n-balls and a new n-ball β

with principal curvature κ_i is added, the new average principal curvature $\bar{\kappa}_i^{n+1}$ of s is computed as

$$\bar{\kappa}_i^{n+1} = (\kappa_i + n\bar{\kappa}_i^n)/(n+1) \qquad \text{for } i = 1, 2.$$

If instead β is deleted, the new average principal curvature $\bar{\kappa}_i^{n-1}$ of s is computed as

$$\bar{\kappa}_i^{n-1} = (n\bar{\kappa}_i^n - \kappa_i)/(n-1) \qquad \text{for } i = 1, 2.$$

When the local geometry estimate of an n-ball is updated, the average principal curvatures of the according segments are updated by deleting the n-ball with the old geometry and adding the n-ball with the new geometry.

Maximum curvature direction criterion

Two cylindrical segments with the same average principal curvatures do not necessarily belong to the same cylinder. If such cylindrical segments are close to each other, the principal curvature criterion is not sufficient to discriminate these segments.

The principal curvature direction u_1 of the maximum principal curvature κ_1 of a cylinder is perpendicular to the axis of the cylinder. Thus, the angle between the maximum principal curvature directions is used to discriminate close cylindrical segments. Two n-balls β_1 and β_2 are clustered to the same cylindrical segment, if the angle between the projection of their maximum principal curvature directions to the plane $\boldsymbol{n}_{\beta_1} + \boldsymbol{n}_{\beta_2}$ is less than a threshold ε_{cd}.

4.2.3 On-line region growing

N-balls are clustered into segments based on their surface type and the segmentation criteria using region growing. The region growing processes the data stream on-line. This implies two requirements for region growing:

- The n-balls are received by the region growing in an arbitrary order, because the raw points are streamed in an arbitrary order, the implementation is parallelized, and the operator might pause the scan process and continue at an arbitrary new region of the object.

- The local geometry estimates of an n-ball can change, because of multiple scans of the same object region. It can also happen that the local surface properties of an existing n-ball change in such a way, that the ball's basic surface type changes. In this case, the n-ball needs to be assigned to another region or represent a new region. Thus, segments can merge or split at any time and all affected n-balls must be re-assigned.

When an n-ball is received by the region growing, it is either added, deleted or updated.

Surface type	Threshold	Minimized quantity
flat	$\varepsilon_{d_\perp}$, perpendicular distance	normal angle
ridge, valley	ε_{cd}, curvature direction angle	principal curvature difference
peak, pit		mean curvature difference
other		normal angle

Table 4.2: Segmentation according to segmentation criteria.

Adding n-balls

Every new n-ball is initially assigned to a new segment. Then, the region growing starts from this n-ball analyzing its immediate neighbors. These are clustered into a common segment depending on the surface type and segmentation criteria as shown in Table 4.2. N-balls with surface types saddle ridge, saddle valley, or minimal surface are clustered into free-form segments.

The analysis of the neighboring n-balls either yields a most similar n-ball or no similar n-ball. In the latter case, the new n-ball keeps its new segment. In the former case, the segments of the new and its most similar n-ball are merged, where the n-balls of the segment with less n-balls are assigned to the other segment and the smaller segment is deleted.

Deleting n-balls

To delete an n-ball, it is removed from its segment. If it was the last n-ball of this segment, the segment is also deleted.

Updating n-balls

To update the local geometry of an n-ball, it is removed from its actual segment and a new temporary segment is generated. For this new temporary segment the region growing is triggered. Thus, the changed n-ball is assigned to the segment with the most similar geometry. This can be the current segment again or a neighboring segment that now fits better because of the updated properties of the n-ball.

4.2.4 Segment type detection

Initially, the surface type of a segment is determined by the surface type of its first n-ball. When the segment contains at least ten n-balls, the surface type is determined based on the average principal curvature of all its n-balls. The main reason for introducing this second classification method is due to the fact that spheres or cylinders of large radii are misclassified as planes when only relying on the initial classification. This is because of the thresholds ε_K and ε_H of the Gaussian and mean curvature. Smaller thresholds would result in a misclassification of balls on a plane as non-planar due to measurement noise in the raw point coordinates.

Average curvature criterion	Segment type
$\lvert\bar{\kappa}_1\rvert, \lvert\bar{\kappa}_2\rvert \leq \varepsilon_{PC}$	planar
$\lvert\bar{\kappa}_1\rvert \leq \varepsilon_{PC} \ \oplus \ \lvert\bar{\kappa}_2\rvert \leq \varepsilon_{PC}$	cylindrical
$\lvert\bar{\kappa}_1\rvert, \lvert\bar{\kappa}_2\rvert > \varepsilon_{PC} \ \wedge \ \lvert\bar{\kappa}_1 - \bar{\kappa}_2\rvert \leq \varepsilon_{PC}$	spherical
$\lvert\bar{\kappa}_1\rvert, \lvert\bar{\kappa}_2\rvert > \varepsilon_{PC} \ \wedge \ \lvert\bar{\kappa}_1 - \bar{\kappa}_2\rvert > \varepsilon_{PC}$	ellipsoidal
none of the above	free-form

Table 4.3: Segment types determined by average principal curvature, using the *exclusive or* operator $\oplus$.

Table 4.3 shows the criteria for segment type classification based on the average principal curvatures $\bar{\kappa}_1$ and $\bar{\kappa}_2$. The threshold ε_{PC} is used for numerical reasons. Because the average curvatures are more stable than n-ball curvatures, ε_{PC} is smaller than ε_K and ε_H. It does not matter in which direction the segment is curved, only absolute values are considered.

A minimum size of ten balls for the classification based on mean curvature seems reasonable. Smaller segments are only classified by the fundamental surface type of the balls they contain. It is guaranteed by the region-growing algorithm that only balls of the same surface type can be together in one segment.

4.2.5 Results

Experimental results of the on-line segmentation and reconstruction for different physical models are provided. The experimental results show the strengths and weaknesses of the presented on-line approach.

For the experiments in this Section recorded scans from two different hand-held laser scanners were used. Most scans were created with a hand-held laser scanner using an optical tracking device, see Figure 2.3b on Page 19. This optical tracking causes additional noise and reduces the repetition accuracy.

For the scan of the casket in Figures 4.3c, 4.4c, and 4.5c, a measurement arm based hand-held laser scanner with high precision and low noise was used, see Figure 2.3a on Page 19.

Thresholds

During the experiments different values for the thresholds described in Section 4.2.2 were tested. Because the algorithms will be a part of automated reverse engineering processes, an extensive calibration phase before each scan is not desirable. Thus, for a comparison, the thresholds in Table 4.4 were used for all experiments. The first three thresholds in Table 4.4 are most critical, because they directly influence the surface type classification of the n-balls and the segment type classification based on principal curvature.

Symbol	Threshold	Value
ε_K	Gaussian curvature	0.03
ε_H	Mean curvature	0.07
ε_{PC}	Principal curvature	0.015
ε_n	Plane normal angle	5°
ε_q	Plane normal quality	0.5
$\varepsilon_{d_\perp}$	Perpendicular distance	0.5
ε_s	Principal curvature of spheres	0.2
ε_c	Principal curvature of cylinders	0.2
ε_{cd}	Maximum curvature direction	8°

Table 4.4: The thresholds used for the presented results of the segmentation.

N-ball classification

Different objects were scanned to test the on-line segmentation and reconstruction. The following discussion and the screen shots illustrate the performance of these algorithms while focusing on critical aspects.

The surface types of the n-balls determined by the Gaussian and mean curvature are shown in Figure 4.3. The raw points are also shown, colored according to their n-ball's type. In order to cope with the scan noise, relatively large values are chosen for ε_K and ε_H. This is observable at places with high noise. This leads to a false classification in Figure 4.3 of the cylinders and the cone as planes.

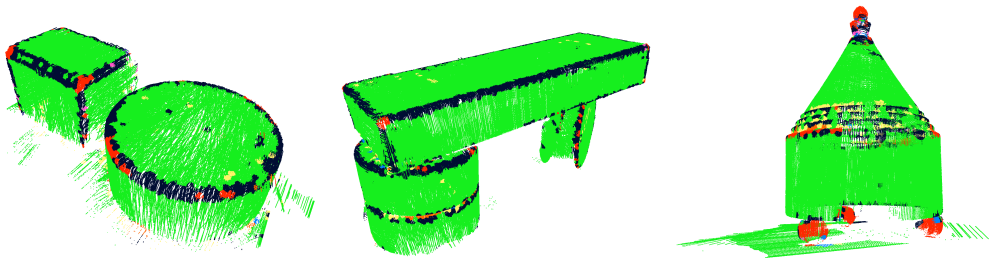
Segment type detection

Figure 4.4 shows the surface type classification of the segments. All cylinders and the cone are correctly classified.

Figure 4.4b shows that the classification of elliptical parts might be unstable. The left part of the cylinder's top blend is classified as cylindrical (blue) and not as ellipsoidal (pink) as on the front part of the cylinder.

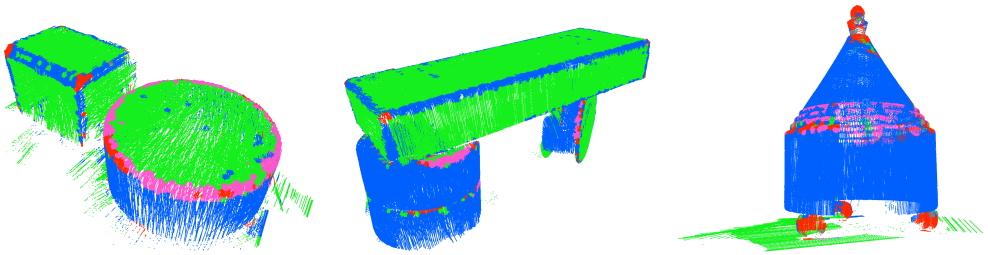
The upper conical part of the casket in Figure 4.4c is classified as cylinder. A cone can not be discriminated from the cylinder by analyzing the principal curvatures. A possible solution would be to analyze the Gauss map, see Weber et al. (2010); Wang and Yu (2011).

The individual segments are shown in Figure 4.5. Narrow areas, i.e., the cylindrical blends in Figure 4.5a, are subdivided into multiple segments. They are so narrow that misclassification of single n-balls can disconnect the segments. On the other hand, the two cylinders in Figure 4.5b are represented by a single segment. N-balls of both object regions are connected by neighborhood links. This is a typical example where the operator sees that the reconstruction is unsatisfactory and additional scan passes are required.



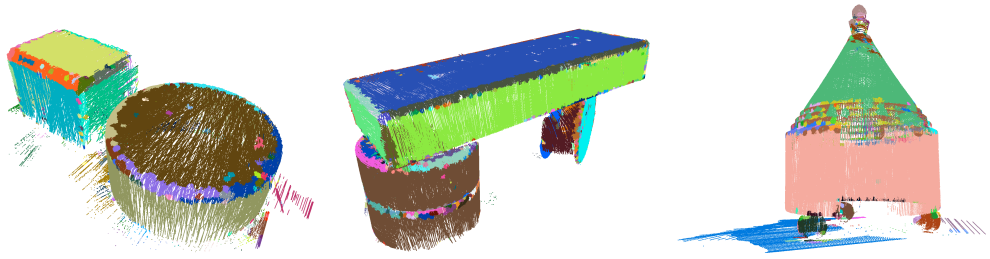
(a) (b) (c)

Figure 4.3: Surface types of the n-balls according to Section 4.2.1. Raw points are displayed and colored according to parent n-ball. Colors: plane, ridge, valley, peak, pit, saddle ridge, saddle valley, minimal surface.



(a) (b) (c)

Figure 4.4: Segment type according to Section 4.2.4. Colors: planar, cylindrical, spherical, ellipsoidal, free-form.



(a) (b) (c)

Figure 4.5: Resulting segments colored individually.

4.3 Quadric Fitting

After the n-balls are segmented, geometric primitives are reconstructed for each segment. The quadric fitting explained in this Section has been published in Denker et al. (2013).

4.3.1 Definition

In order to reduce the memory consumption and the complexity of the representation implicit quadrics are used, similar to Vančo et al. (2008). Using this representation the set of n-balls for a segment can be reduced to a set of ten coefficients defining a quadric

$$0 = q(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{Q}\, \boldsymbol{x} + 2\boldsymbol{P}^T \boldsymbol{x} + R \qquad \text{for } \boldsymbol{x} \in \mathbb{R}^3$$

where

$$\boldsymbol{Q} = \begin{bmatrix} A & D & E \\ D & B & F \\ E & F & C \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{P} = \begin{bmatrix} G \\ H \\ I \end{bmatrix}.$$

4.3.2 Fitting

In order to fit a quadric surface to a segment an algebraic approach is used. Due to the classification of the segments this algebraic fit converges robustly.

A segment consists of n-balls $\beta_1, \ldots, \beta_n$. To fit a quadric surface to this segment the least square solution of the over-determined system of equations

$$q(\boldsymbol{v}_{\beta_i}) = 0, \qquad \text{for } i = 1, \ldots, n \tag{4.1}$$

is computed.

4.3.3 Additional conditions

To improve the robustness of this fit the additional conditions $\nabla_{\boldsymbol{n}} q = 1$ and $\nabla^2_{\boldsymbol{n}} q = 0$ are used, where ∇_n is the directional derivative with respect to the normal direction $\boldsymbol{n}$. This induces a nice behavior outside the zero set of the quadric. Otherwise the overall size of the coefficients would be arbitrary and tend towards extreme large or small values that may cause numerical problems. The additional functions also ensure that the quadric is positive outside the surface and negative inside. This leads to the additional equations

$$\nabla_{\boldsymbol{n}_{\beta_i}} q(\boldsymbol{v}_{\beta_i}) = 1, \qquad \text{for } i = 1, \ldots, n \qquad \text{and} \tag{4.2}$$

$$\nabla^2_{\boldsymbol{n}_{\beta_i}} q(\boldsymbol{v}_{\beta_i}) = 0, \qquad \text{for } i = 1, \ldots, n. \tag{4.3}$$

These two conditions control the size of the coefficients of q and ensure that q is positive in direction of the normal. For the fit a linear combination of (4.1), (4.2),

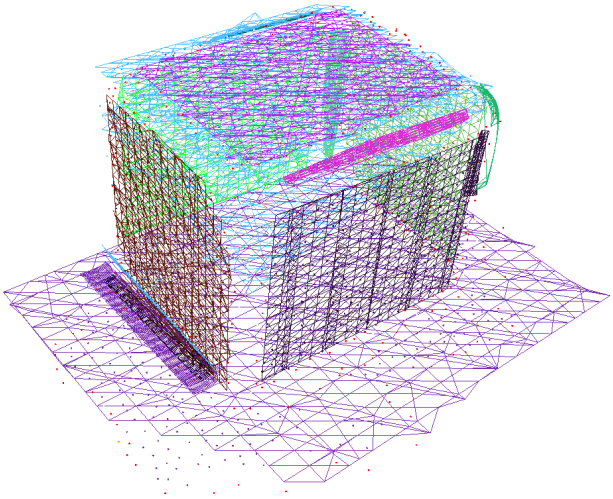


Figure 4.6: Marching cubes visualization of reconstructed quadrics.

and (4.3) is solved using *singular value decomposition* (SVD).

4.3.4 Planar fitting

Using the above quadric fitting for planes might result in a double plane due to noise in the data. However, for a plane no complete set of quadric parameters is necessary. Setting $Q = 0$ in (4.1) and (4.2) simplifies the fitting for planes.

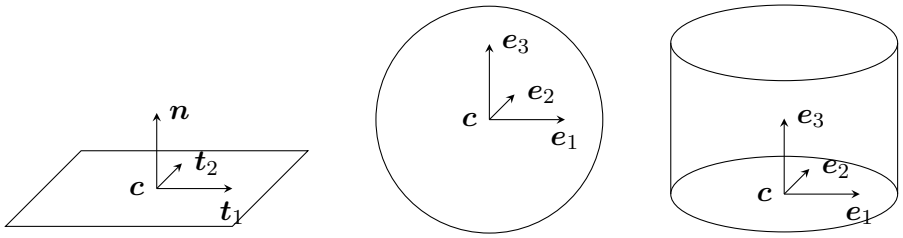
Since the set of geometric primitives that could be represented by the segments also includes simple planes, the previously described algebraic approach needs a little correction. Due to the possibility of the algebraic approach to fit a double plane to the neighborhood balls and due to the naturally included noise of the data gained by a laser scanner, the algebraic fit would always prefer the double plane over the regular plane. Using the classification of the segments it is possible to determine whether the surface is a plane or not before the algebraic fitting step. In order to prevent the fitting of double planes, in that case a simplified plane fitting approach is used.

4.3.5 Results

Figure 4.6 shows a simple marching-cubes visualization of the quadrics fitted to the scan of a wooden cuboid.

4.4 Geometric Primitives from Quadrics

Geometric primitives in CAD systems are usually represented by a position, a rotation, and a set of parameters defining the primitive. In this Section, the computation



(a) plane (b) sphere (c) cylinder

Figure 4.7: Canonical system of quadrics of geometric primitives.

of all these parameters from quadrics is described. This method has been published in Denker et al. (2013).

4.4.1 Canonical system

Quadrics are usually represented in normal form. For this the principal axes of the quadric and characteristic parameters of the surface (e.g., radii, etc.) are computed.

The canonical system of a quadric is given by the eigenvectors $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, $\boldsymbol{e}_3$ of $\boldsymbol{Q}$. To get the correct translation of the quadric surface, the equation $\boldsymbol{Qx} + \boldsymbol{P} = \boldsymbol{0}$ is solved. The solution of this equation yields either a center point, a point on a centerline, or a point on a center plane. This yields a complete representation of the spatial position of the quadric surface, see Figure 4.7.

4.4.2 Degenerated quadrics

For some degenerated quadrics (e.g., planes) this approach fails. Furthermore, to compute the characteristic parameters of a quadric surface special rules apply. Which rules apply is determined by the segment type. These rules are different for planes, spheres, cylinders, and cones.

Plane

Since for a plane $\boldsymbol{Q}$ is singular the computation of the spatial position is different. The translational component is determined by the position of a center point $\boldsymbol{c}$, which is the mean of all points projected onto the surface. A possible choice for the canonical system is the normal $\boldsymbol{n}$ of the segment and any suitable 2d system in the plane, see Figure 4.7a.

Sphere

Since the quadric representation of a sphere has a single center the translation of the sphere can be computed directly from its center point $\boldsymbol{c}$, see Figure 4.7b. The radius of the sphere is computed by ray casting the axes of the canonical system from $\boldsymbol{c}$. To compute the intersections the ray equation is substituted into q.

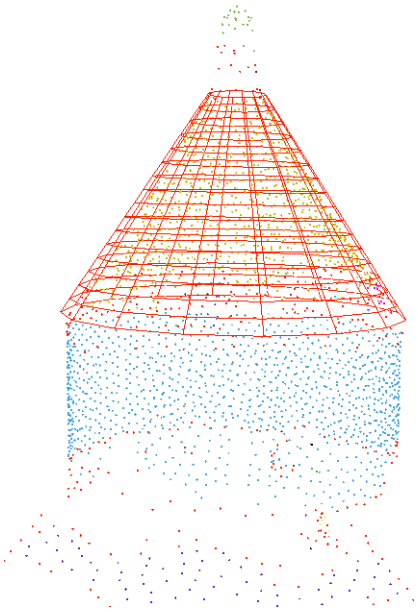


Figure 4.8: Cone reconstructed from a scan of a casket.

Cylinder and cone

The solution of $\boldsymbol{Qx} + \boldsymbol{P} = 0$ for the cylinder yields an arbitrary point on the center line of the cylinder. This center line has direction $\boldsymbol{e}_3$, see Figure 4.7c. Projecting all vertices of boundary n-balls onto $\boldsymbol{e}_3$ yields two extremal projected points. The planes through these points normal to $\boldsymbol{e}_3$ are the upper and lower cap of the cylinder. For the translational component, the intersection of the bottom cap with the center line gives a center point $\boldsymbol{c}$ of the cylinder. The distance of the caps is the height h of the cylinder.

To estimate the radius of the cylinder the distance of $\boldsymbol{c} + h\boldsymbol{e}_3$ to the cylinder surface is computed using ray casting in direction $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$.

For a cone the computations are the same except that the radii are computed in the top and bottom cap.

4.4.3 Results

The methods for the primitive reconstruction are able to reconstruct the described geometric primitives. Figure 4.8 shows the reconstruction of a cone from the scan of a casket. In Figure 4.9 the reconstruction of a cylinder mantle and its planar caps is demonstrated with a scan of a wooden cylinder. The accuracy of the geometric model depends on the accuracy of the implicit quadric surface.

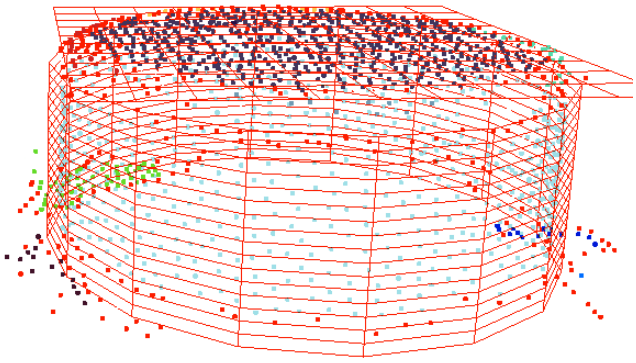


Figure 4.9: Geometric primitives reconstructed from a scan of a wooden cylinder.

4.5 Local Sharp Feature Detection with IRWLS

The quadratic polynomial P_β approximates the geometry of the raw points of an n-ball β accurately only when the geometry is sufficiently smooth, see Section 4.1.2. Sharp features cannot be approximated accurately by a single quadratic polynomial. Thus, two polynomial approximations are used to reconstruct sharp features.

To compute P_β a *least absolute deviation fit* is used instead of a least squares fit. It is based on the L^1 norm instead of the L^2 norm. For the computation *iteratively re-weighted least squares* (IRWLS) (Holland and Welsch, 1977) are used. For IRWLS the weighted least squares fitting with weights $w_i = 1/\sqrt{d_i}$ is computed, where d_i is the distance of the fit to the raw points. These weights are adjusted in every iteration. The weight adjustment compensates for the difference between the L^1 and L^2 norm. Usually, after approximately five iterations P_β is sufficiently close to a least absolute deviation fit.

During the IRWLS computation the number of outliers with distance $d_i > \varepsilon_{\text{sd}} r$ is recorded, for the threshold ε_{sd} see Table 4.5 on Page 86. If there are more than 20% outliers, a sharp feature is assumed and a second polynomial P^2_β is fit. This second polynomial is initially IRWLS fitted to the outliers. Then, both polynomials $P^1_\beta = P_\beta$ and P^2_β are IRWLS fitted to all raw points with weights

$$w^1_i = \max\left(d^2_i - \varepsilon_{\text{sd}}, 0\right) / \sqrt{d^1_i},$$
$$w^2_i = \max\left(d^1_i - \varepsilon_{\text{sd}}, 0\right) / \sqrt{d^2_i}.$$

Thus, raw points that are close to one of the fits obtain a small weight for the other fit. Note, that the superscript k refers to quantities related to $P^k_\beta, k = 1, 2$.

The ratio of outliers of both surfaces to the number of fitted raw points defines a quality score q_{ls} for the fit. For sharp features, the arithmetic mean of the raw points $\boldsymbol{v}_\beta$ is projected onto the intersection curve of P^1_β and P^2_β by iteratively projecting it

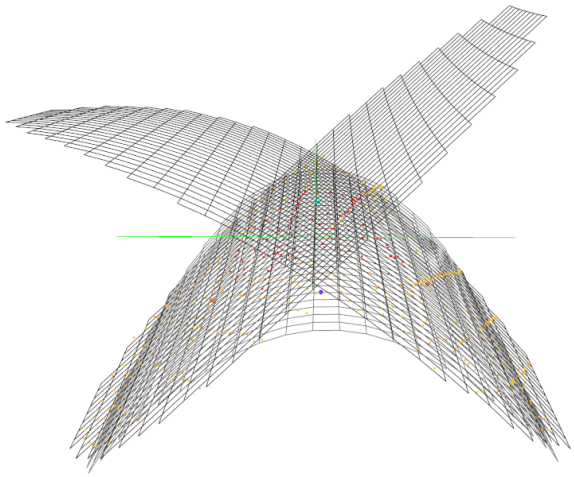


Figure 4.10: Visualization of the quadratic polynomials P_β (gray), P_β^1, and P_β^2 (black) of an n-ball on an edge. Raw points inside (red) and outside (yellow) the n-ball within a certain radius are fitted. The mean point $\boldsymbol{v}_\beta$ (violet) is iteratively projected onto the sharp feature (cyan).

to the intersection line of local tangent planes.

Figure 4.10 shows a local visualization of the surfaces fitted for the sharp feature detection. The resulting geometry is displayed in Figure 4.11. The n-balls near the edge have two different normals and the points $\boldsymbol{p}$ are projected onto the feature line.

4.6 Accumulated Means

In on-line computations as the on-line reconstruction, means allow an increased stability. Means over a segment are much more stable than purely local geometric properties. Thus, means will be used for segmentation and reconstruction in the subsequent Sections.

An incremental computation of means is introduced in Welford (1962); van Reeken (1968). These methods are compared with other non-incremental methods to compute means and variances with respect to numerical performance in Ling (1974); Chan et al. (1983). The application of these methods for variance computations is also discussed in Knuth (1998). These incremental means are constructed to add single values to the mean. The removal of values or merging of multiple means is not considered.

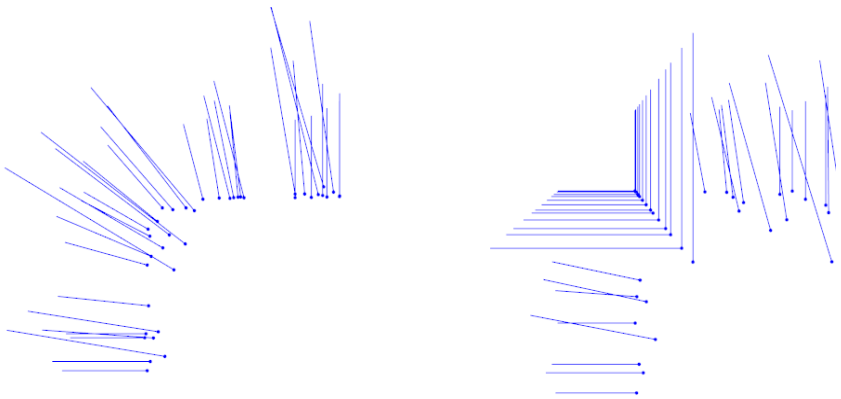


Figure 4.11: N-ball points $\boldsymbol{p}$ and normals $\boldsymbol{n}$ near a sharp edge computed without and with sharp feature detection.

Clustering methods like k-means (Press et al., 2007) or *unweighted pair group method with arithmetic mean* (UPGMA) (Sokal and Michener, 1958) use mean positions of multi-dimensional data. On-line k-means methods as Zhong (2005) allow the incremental addition of data to the mean centroids of the clusters. New values are added with a fixed or decreasing learning rate. Data changing the cluster is not removed from the old cluster centroids. Note, that on-line k-means is not on-line in the sense defined earlier. Efficient implementations of UPGMA, as in Gronau and Moran (2007) based on arithmetic distance means, use a reduction formula to merge means. Though, no higher order geometric properties are used. The advantage of using means of geometric properties instead of purely local geometric properties for the detection of surface segments is the increased stability in the computations.

A method that combines local geometric properties to more stable estimations is presented in Page et al. (2002). Discrete estimates for normals and curvatures are combined over a local area using a voting algorithm. However, this method is not on-line and no geometric parameters derived from the local geometry are used.

4.6.1 Arithmetic means

Weighted and unweighted arithmetic means of a set of n data x_i with constant weights w_i are defined as

$$\bar{x}_n = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i} \quad \text{and} \quad \tilde{x}_n = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

For on-line computations means are computed by incremental addition of data. Besides, it is necessary to remove data from a mean or to merge two means. A quantity is called *accumulated* when it supports these three *accumulation operations*, e.g., $\bar{x}_n$ is an accumulated weighted arithmetic mean.

4.6.2 Accumulated arithmetic means

The incremental computation of an unweighted arithmetic mean $\tilde{x}_n$ when adding a value $x_n, n > 1$, is defined as

$$\tilde{x}_n = \tilde{x}_{n-1} + \frac{1}{n}\left(x_n - \tilde{x}_{n-1}\right), \qquad \tilde{x}_1 = x_1.$$

For numerical reasons it is avoided to add $(n-1)\tilde{x}_{n-1}$, see Ling (1974). To compute a weighted arithmetic mean $\bar{x}_n$, the average weight $\tilde{w}_n$ is used for accumulation operations

$$\bar{x}_n = \bar{x}_{n-1} + \frac{w_n}{(n-1)\tilde{w}_{n-1} + w_n}\left(x_n - \bar{x}_{n-1}\right), \qquad \bar{x}_1 = x_1.$$

Similarly, a value $x_{n+1}, n \geq 1$, can be removed from an unweighted arithmetic mean by

$$\tilde{x}_n = \tilde{x}_{n+1} - \frac{1}{n}\left(x_{n+1} - \tilde{x}_{n+1}\right)$$

and from a weighted arithmetic mean by

$$\bar{x}_n = \bar{x}_{n+1} - \frac{w_{n+1}}{(n+1)\tilde{w}_{n+1} - w_{n+1}}\left(x_{n+1} - \bar{x}_{n+1}\right).$$

Two arithmetic means $\tilde{a}_n$ and $\tilde{b}_m$ with $m \leq n$ are merged to one combined arithmetic mean $\tilde{x}_{n+m}$ by

$$\tilde{x}_{n+m} = \tilde{a}_n + \frac{m}{n+m}\left(\tilde{b}_m - \tilde{a}_n\right). \tag{4.4}$$

The combined weighted arithmetic mean is computed using the arithmetic means of the weights $\tilde{u}_n$, $\tilde{v}_m$ of the partial weighted arithmetic means $\bar{a}_n$ and $\bar{b}_m$

$$\bar{x}_{n+m} = \bar{a}_n + \frac{m\tilde{v}_m}{n\tilde{u}_n + m\tilde{v}_m}\left(\bar{b}_m - \bar{a}_n\right).$$

The weight mean $\tilde{w}_{n+m}$ for the combined mean $\bar{x}_{n+m}$ is computed from $\tilde{u}_n$, $\tilde{v}_m$ by Equation (4.4). Note, that the computational complexity of all accumulation operations is $O(1)$, which is optimal for an on-line algorithm.

4.6.3 Accumulated arithmetic means in the ball tree

For each n-ball in a ball tree sets of values are obtained, that are combined in accumulated arithmetic means. Unweighted arithmetic means are used for position $\boldsymbol{p}$, normal $\boldsymbol{n}$, and ball radius r, see Section 4.1.2. Their means $\tilde{\boldsymbol{p}}$, $\tilde{\boldsymbol{n}}$, $\tilde{r}$ are used to detect planar segments. Accumulated weighted arithmetic means $\bar{\kappa}_1$, $\bar{\kappa}_2$, $\bar{H}$ are computed for principal curvatures κ_1, κ_2, and mean curvature H. The respective weights used in the computation are given by the quality score q_{ls} of the least squares fit, see Section 4.5.

4.6.4 Indirect accumulated means

Accumulated weighted arithmetic means for principal curvatures and mean curvature can be used to compute *indirect accumulated weighted means* for the hypothetical cylinder radii $\bar{r}_1$, $\bar{r}_2$ and the hypothetical spherical radius $\bar{r}_s$

$$\bar{r}_i = -\bar{\kappa}_i^{-1}, i = 1, 2, \quad \text{and} \quad \bar{r}_s = -\bar{H}^{-1}.$$

Other geometric quantities are defined by normalized, un-oriented, vector-valued data, e.g., principal curvature directions $\boldsymbol{d}_1$, $\boldsymbol{d}_2$. Thus, arithmetic means cannot be used to combine principal directions. Due to normalization, this data lies on the unit-sphere. Their distribution on the unit-sphere is estimated by a PCA step.

For both principal curvature directions $\boldsymbol{d}_i, i = 1, 2$, the 3×3 co-variance matrix $\boldsymbol{C}_i$ is computed. The eigenvalues $\lambda_{i,j}, j = 1, 2, 3$, of $\boldsymbol{C}_i$ characterize these distributions. The eigenvectors of the largest eigenvalues $\lambda_{i,3}$ are used as unweighted means $\tilde{\boldsymbol{d}}_i$ for the un-oriented directions. Co-variance matrices $\boldsymbol{C}_i$ are accumulated, because data $\boldsymbol{d}$ can be added or removed by

$$\boldsymbol{C}_i^{\text{new}} = \boldsymbol{C}_i^{\text{old}} \pm \boldsymbol{d}_i \cdot \boldsymbol{d}_i^t,$$

and two co-variance matrices $\boldsymbol{C}_i^1, \boldsymbol{C}_i^2$ are merged by

$$\boldsymbol{C}_i = \boldsymbol{C}_i^1 + \boldsymbol{C}_i^2,$$

yielding the co-variance of the union of both sets of un-oriented directions.

Thus, $\tilde{\boldsymbol{d}}_i, i = 1, 2$, are *indirect accumulated means*. They provide information for cylinder detection: the cylinder axis has the same direction as the principal direction of the smaller absolute principal curvature.

The complexity for the accumulation operations for a 3×3 co-variance matrix $\boldsymbol{C}$ is $O(1)$. Due to the constant size of $\boldsymbol{C}$, the computation of its eigenvalues and eigenvectors has complexity $O(1)$. So, the overall complexity is $O(1)$.

4.6.5 Accumulated means depending on means

For each segment hypothetical cylinder and sphere center points are estimated by locally approximated center points

$$\boldsymbol{p}_i = \boldsymbol{p} - \bar{r}_i \cdot \boldsymbol{n}, i = 1, 2, \quad \text{and} \quad \boldsymbol{p}_s = \boldsymbol{p} - \bar{r}_s \cdot \boldsymbol{n}.$$

These local estimates accumulate to the weighted arithmetic means $\bar{\boldsymbol{p}}_1$, $\bar{\boldsymbol{p}}_2$, and $\bar{\boldsymbol{p}}_s$ with weights $|\bar{\kappa}_1|$, $|\bar{\kappa}_2|$, and $\left|\bar{H}\right|$, respectively. Weighted arithmetic means are used to account for the non-linear distribution of center points and the fact that there is no multi-dimensional harmonic mean. Note, that the local estimates and the weights vary with time and need to be stored for the accumulation operations.

With $\bar{\boldsymbol{p}}_1$, $\bar{\boldsymbol{p}}_2$, and $\bar{\boldsymbol{p}}_s$ a more stable local estimate for the radii can be computed,

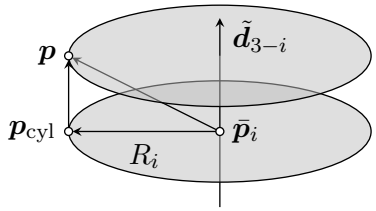


Figure 4.12: Points and directions of a cylindrical segment.

see Figure 4.12,

$$R_i = \|(\boldsymbol{p} - \bar{\boldsymbol{p}}_i) - ((\boldsymbol{p} - \bar{\boldsymbol{p}}_i)\tilde{\boldsymbol{d}}_{3-i})\tilde{\boldsymbol{d}}_{3-i}\|, \qquad i = 1, 2,$$
$$R_s = \|\boldsymbol{p} - \bar{\boldsymbol{p}}_s\|.$$

The accumulated weighted means of corrected radii $\bar{R}_1$, $\bar{R}_2$, and $\bar{R}_s$ are computed again with weights $|\bar{\kappa}_1|$, $|\bar{\kappa}_2|$, and $\left|\bar{H}\right|$.

4.6.6 Scores for geometric primitives

For segmentation, scores are defined to measure the fitting quality of an n-ball to planar, spherical, cylindrical, or free-form geometry. Multiplicative scores are used that are centered at 1 using a normalization weight ω, see Table 4.5 on Page 86. Values smaller that 1 denote high fitting quality. Partial scores s_i are combined by weighted multiplication using weights w_i to yield a combined score

$$s = \prod_{i=1}^{n} \left((s_i - 1)\, w_i + 1\right) \text{ with } w_i \in (0, 1).$$

Note, that for each segment all scores are maintained at all times. Currently, scores for planar, cylindrical, spherical and free-form geometries are computed. Each of these scores consists of a partial score for the segment size, a distance score, an angle score, and a curvature score. The latter is used only for cylindrical and spherical geometries. In the computations all means belong to the same segment and all local data belongs to the corresponding n-ball. The weights used for the multiplicative combination are 3/4 for the distance and angle sores and 1/2 otherwise.

Planar segments

The score for planar segments s_{pln} is computed based on:

Distance score: Normal distance $\tilde{\boldsymbol{n}}(\tilde{\boldsymbol{p}} - \boldsymbol{p})/(\bar{r}\omega_{\text{nd}})$.

Angle score: $\sphericalangle(\boldsymbol{n}, \tilde{\boldsymbol{n}})$.

Size score: Reciprocal number of n-balls in the segment.

Cylindrical segments

The score for cylindrical segments s_{cyl} is based of the direction of the cylinder curvature $\bar{\kappa}_\delta$ that is orthogonal to the cylinder axis $\tilde{\boldsymbol{d}}_\Delta$ with

$$\delta = \underset{i=1,2}{\operatorname{argmin}} |\lambda_{i,3}| \quad \text{and} \quad \Delta = \underset{i=1,2}{\operatorname{argmax}} |\lambda_{i,3}|$$

with the largest eigenvalue $\lambda_{i,3}$ of the co-variance matrix $\boldsymbol{C}_i$, as defined in Section 4.6.4. Thus, the index δ is an indirect accumulated quantity characterizing the type of the cylinder.

To simplify radius computations a point on the cylinder surface in the plane normal to $\tilde{\boldsymbol{d}}_\Delta$ containing $\bar{\boldsymbol{p}}_\delta$ is computed, see Figure 4.12,

$$\boldsymbol{p}_{\text{cyl}} = \boldsymbol{p} - \left((\boldsymbol{p} - \bar{\boldsymbol{p}}_\delta)\, \tilde{\boldsymbol{d}}_\Delta \right) \tilde{\boldsymbol{d}}_\Delta .$$

The partial scores for cylindrical segments are:

Distance score: Radius difference $\frac{\left| \|\boldsymbol{p}_{\text{cyl}} - \bar{\boldsymbol{p}}_\delta\| - |\bar{R}_\delta| \right|}{\omega_{\text{cyl}} |\bar{R}_\delta|}$.

Angle score: Average of $\sphericalangle(\boldsymbol{d}_\Delta, \tilde{\boldsymbol{d}}_\Delta)$ and $\sphericalangle(\boldsymbol{n}, \boldsymbol{p}_{\text{cyl}} - \bar{\boldsymbol{p}}_\delta)$.

Curvature score: $\max(|\kappa_\delta|, |\bar{\kappa}_\delta|)/\min(|\kappa_\delta|, |\bar{\kappa}_\delta|)$.

Size score: Reciprocal number of n-balls in the segment.

Spherical segments

The score for spherical segments s_{sph} is computed from:

Distance score: Radius difference $\frac{\left| \|\boldsymbol{p} - \bar{\boldsymbol{p}}_s\| - |\bar{R}_s| \right|}{\omega_{\text{sph}} |\bar{R}_s|}$.

Angle score: $\sphericalangle(\boldsymbol{n}, \boldsymbol{p} - \bar{\boldsymbol{p}}_s)$.

Curvature score: $\max(|H|, |\bar{H}|)/\min(|H|, |\bar{H}|)$.

Size score: Reciprocal number of n-balls in the segment.

Free-form segments

For free-form geometry a score s_{ffg} is computed based mostly on local values of two neighbored n-balls β and β_{nb}.

Distance score: Normal distance $\boldsymbol{n}_{\text{nb}}(\boldsymbol{p}_{\text{nb}} - \boldsymbol{p})/(\bar{r}\omega_{\text{nd}})$.

Angle score: $\sphericalangle(\boldsymbol{n}, \boldsymbol{n}_{\text{nb}})$.

Size score: Reciprocal number of n-balls in the segment.

Balancing scores

Since the scores are combined from different geometric quantities, the ranges for the overall scores s_{pln}, s_{cyl}, s_{sph}, and s_{ffg} differ significantly, e.g., good values of s_{pln} tend to be much smaller than good values of s_{cyl}. To compensate for this effect and to avoid a time-consuming re-balancing of the partial scores, the overall scores are re-balanced by correction factors. These factors are 2.5 for s_{pln}, 0.7 for s_{cyl}, 0.9 for s_{sph}, and 6.0 for s_{ffg}.

For s_{pln}, s_{cyl}, and s_{sph} accumulated means $\bar{s}_{\text{pln}}$, $\bar{s}_{\text{cyl}}$, and $\bar{s}_{\text{sph}}$ are computed for each segment. These means are weighted by the quality q_{ls} and are used to determine the overall segment type. The minimum value of these mean scores determines the segment type. If the minimum is larger than 1 the segment is classified as free-form geometry.

4.7 Segmentation and Reconstruction with Accumulated Means

A new segmentation algorithm based on accumulated means is presented in this Section. In contrast to the method in Section 4.2, the segmentation and reconstruction are not separated. The parameters of the reconstructed geometric primitives are accumulated means that are computed and used for segmentation.

The segmentation algorithm is implemented as a separate layer on top of the ball tree described in Section 4.1. N-balls can be added, removed or modified at any time. For each of these updates the segmentation needs to be updated.

4.7.1 Changing single n-balls

When a new n-ball is generated, it is assigned to the best-fitting segment. To find the best-fitting segment the score s_{pln}, s_{cyl}, s_{sph}, or s_{ffg} of this n-ball for each hypothetical segment that contains an n-ball in the local neighborhood is computed, see Section 4.6.6. The segment type of the hypothetical segment determines which score for the n-ball is computed. Then it is added to the segment with the minimal score by adding its data and scores to the means of the segment. The n-ball is also inserted to the segment data structures.

To remove an n-ball its old values are removed from the means of its segment. Then, the n-ball is removed from the segment data structure. If a segments contains no n-balls after a removal, the segment will be deleted.

For modifications of n-balls the same minimal scores as for adding n-balls are computed. When a score indicates a change of segments, the n-ball is removed from the old segment and added to the new one. Otherwise, the old values of the n-ball are removed from the means of its segment and the new values are added.

Accumulating the means of a segment is of complexity $O(1)$. Assuming that the size of the local neighborhood of an n-ball is constant, the search for the best-fitting segment has also complexity $O(1)$. Updating the set of n-balls of a segment has

complexity $O(1)$ and updating the global hash table of n-balls has also complexity $O(1)$, see Section 4.1.3.

4.7.2 Merging segments

After several updates for one segment, a hypothetical merged segment with each of its neighbor segments is computed. This hypothetical segment contains the data of both segments using the accumulation formula for the means and co-variance matrices. Note, that in the sequel quantities with superscript $k = 1, 2$ refer to data associated with one of the merged segments and quantities without superscript to data of the resulting hypothetical segment.

The segment type of the hypothetical segment is determined using the accumulated mean scores $\bar{s}_{\text{pln}}$, $\bar{s}_{\text{cyl}}$, $\bar{s}_{\text{sph}}$. Depending on the resulting segment type further merge conditions based on pre-defined thresholds ε is evaluated, see Table 4.5.

Planar segments

For planar segments the normal distance between the center points $\tilde{\boldsymbol{p}}^1$, $\tilde{\boldsymbol{p}}^2$ of both original segments

$$|\tilde{\boldsymbol{n}}^1(\tilde{\boldsymbol{p}}^1 - \tilde{\boldsymbol{p}}^2)| + |\tilde{\boldsymbol{n}}^2(\tilde{\boldsymbol{p}}^2 - \tilde{\boldsymbol{p}}^1)| < \varepsilon_{\text{nds}}(\tilde{r}^1 + \tilde{r}^2)$$

and the angle between their normals $\tilde{\boldsymbol{n}}^1$, $\tilde{\boldsymbol{n}}^2$

$$\sphericalangle(\tilde{\boldsymbol{n}}^1\tilde{\boldsymbol{n}}^2) < \varepsilon_{\sphericalangle}$$

are evaluated.

Cylindrical segments

For cylindrical segments the angle between the cylinder axes

$$\sphericalangle(\tilde{\boldsymbol{d}}^1_\Delta\tilde{\boldsymbol{d}}^2_\Delta) < \varepsilon_{\text{ca}},$$

the difference of radii

$$|\bar{R}^1_\delta - \bar{R}^2_\delta| < \varepsilon_{\text{cr}}|\bar{R}_\delta|,$$

and the deviation of the cylinder axes

$$\|(\bar{\boldsymbol{p}}^2_\delta - \bar{\boldsymbol{p}}^1_\delta) - ((\bar{\boldsymbol{p}}^2_\delta - \bar{\boldsymbol{p}}^1_\delta)\tilde{\boldsymbol{d}}_\Delta)\tilde{\boldsymbol{d}}_\Delta\| < \varepsilon_{\text{cc}}|\bar{R}_\delta|.$$

are evaluated.

Spherical segments

For spherical segments the difference of radii

$$|\bar{R}_s^1 - \bar{R}_s^2| < \varepsilon_{\mathrm{sr}} |\bar{R}_s|$$

and the distance of their center points

$$\|\bar{\boldsymbol{p}}_s^1 - \bar{\boldsymbol{p}}_s^2\| < \varepsilon_{\mathrm{sc}} |\bar{R}_s|$$

are evaluated.

Free-form segments

Free-form segments cannot be merged.

Merge operation

When the hypothetical segment satisfies the conjunctive combination of the respective merge conditions, the merge operation is realized: The means and data of the smaller segment are merged to the means and the data of the larger segment, the neighbor information is updated, and then the smaller segment is deleted. After a realized merge, further checks for merges of the involved segments are dropped.

The complexity to compute a hypothetical segment is $O(1)$. Hypothetical segments are computed for each of the k neighbors, so the complexity of checking for merges is $O(k)$. The complexity of a realized merge operation is $O(1)$ for merging the means. Merging the sets of n-balls of the segments has an amortized complexity of $O(m)$, where m is the number of n-balls in the smaller segment. The global hash table associating n-balls to segments is updated with complexity $O(m)$. Thus, the overall complexity to merge two segments is $O(k + m)$.

4.7.3 Consistency checks

Every time a segment changes its type, all its n-balls are checked for consistency. This is also done when a cylindrical segment changes the direction of its axis. For each n-ball in the segment the segment with the minimal score s_{pln}, s_{cyl}, s_{sph}, or s_{ffg} is determined. If it differs from the current segment, the n-ball is removed from the current segment and added to the new segment.

The computational complexity of this consistency check is $O(m)$ where m is the number of n-balls in the segment. Each of these n-balls determines its best-fitting neighboring segment in $O(1)$. Then, it is removed from the current segment in $O(1)$ and added to another one in $O(1)$, if necessary.

4.8 Boundary Reconstruction

In CAD applications most surfaces are trimmed. A boundary curve is constructed that represents the outer contour of the surface. For rendering of the results of the on-line CAD reconstruction a similar approach is used. Polygonal boundaries are constructed that surround the areas of the primitives that are part of the reconstructed segments.

A polygonal boundary B is a ordered sequence of n-balls β_i with $1 \leq i \leq n_B$. By definition, a boundary enclosing a surface is ordered counterclockwise when seen from the outside.

4.8.1 Boundary initialization

Segments are reconstructed using a region growing approach, see Section 4.7. A segment starts with a single n-ball and is extended by adding additional n-balls. When a segment consists of three n-balls, a first initial boundary B with $n_B = 3$ can be constructed. By using the current segment type a local projection is used to choose a counterclockwise orientation of B.

4.8.2 Boundary expansion

All the following operations use a local planar projection based on the segment data, e.g. in the direction towards the center point of a spherical segment. For each n-ball β_{new} that is added, modified or removed from a segment, B is updated. When a newly added n-ball is outside of B, it is inserted to the closest edge $\langle \beta_i, \beta_{i+1} \rangle$ of B.

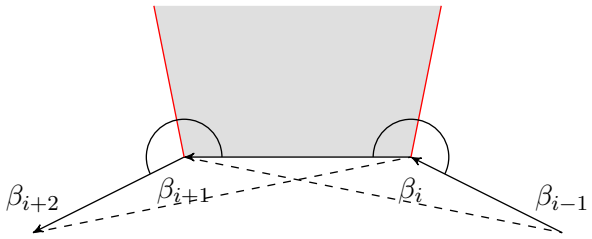


Figure 4.13: Sector (gray area) between half-angle lines (red) of an edge $\langle \beta_i, \beta_{i+1} \rangle$ of the boundary.

One or more neighbors β_i of β_{new} on the boundary B are searched. Both its edges $\langle \beta_{i-1}, \beta_i \rangle$ and $\langle \beta_i, \beta_{i+1} \rangle$ are candidates for inserting β_{new}. In the projection, β_{new} has to lie within the sector bounded by the half-angle lines to its two neighboring edges, see Figure 4.13. The edge that fulfills this condition and whose center point has the smallest distance to β_{new} is chosen for insertion. After the insertion of β_{new}, consistency checks are performed to ensure a nice shape of the boundary.

Changes of the values of an n-ball already on the boundary causes just a call of the consistency checks. If the modified n-ball is not on the boundary, it is tried to

add them to a boundary like a newly added n-ball.

When removing an n-ball from the segment, it is also removed from the boundary. If it was on the boundary, it is tried to re-connect the boundary by inserting neighboring n-balls to the boundary. In the projection space, the neighboring n-ball β_{nb} with the smallest sum of inner angles to the half-angle lines next to the deleted n-ball is searched, see Figure 4.13. If such an β_{nb} is found and it lies outside of the boundary edge it would be inserted to, β_{nb} is inserted to the boundary.

4.8.3 Multiple boundaries

A segment can have multiple boundaries B_i, e.g. a complete cylinder mantle has two boundaries around the caps. When scanning such a cylindrical segment, a single boundary B_1 is created that is split to two boundaries B_1, B_2, when a full scan around the cylinder was completed. Whenever two outsides of the boundary are close to each other, such a split operation is performed.

Merging of segments, see Section 4.7.2, requires a merging of their boundaries. After the segments are merged, all boundaries of the segment are checked for neighboring n-balls on different boundaries of the same segment. If a part of both boundaries is close together and runs in opposite direction, the boundaries are merged. The boundary parts that are close to each other are removed and their end points are connected to create a single boundary.

4.8.4 Local consistency checks

After the connections or the n-balls of B are modified, a local consistency check is performed for a single n-ball β_i. If the outside angle between both its edges $\langle \beta_{i-1}, \beta_i \rangle$ and $\langle \beta_i, \beta_{i+1} \rangle$ is smaller that 150°, β_i is removed from the boundary. For all neighbors β_{nb} of β_i that are on the same boundary, intersections are searched between all connected boundary edges. When two intersections are found, the boundary is split by connecting the two segments before and after the intersections. For one intersection, it is assumed that the shorter part of the boundary is invalid, and it is deleted.

If in the local neighborhood of β_i a point on another boundary on the same segment is found, a merge operation of these boundaries is initiated.

Finally, segments of two edges on B that are close to the edges next to β_i and have opposite directions are found and a split operation is performed, see Section 4.8.3.

4.8.5 Surface rendering with boundaries

To render a shaded surface picture of the result geometry, it is necessary to trim the reconstructed primitives using the boundaries. Otherwise only complete spheres or cylinders could be rendered. The current reconstruction provides planes, cylinders and spheres that need to be trimmed and rendered.

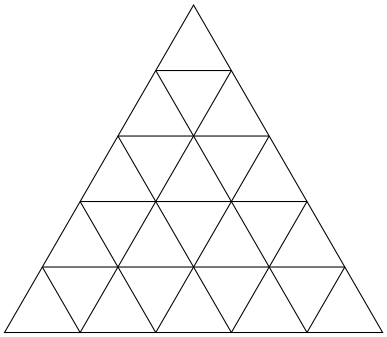


Figure 4.14: Subdivision for rendering sphere and cylinder boundaries.

Planes

For trimming of planes, a tessellation implementation from the *OpenGL Utility Library* (GLU) (OpenGL Architecture Review Board, 1992) is used. The segment and the boundary are mapped into a local two dimensional coordinate system. Then the tessellation engine creates a triangulation of the surface. Inconsistencies, like self-intersections in the boundary can be handled automatically by the tessellation using winding rules. For this work, the default rule is used that fills regions with odd winding numbers.

Spheres

Each single boundary of a sphere is projected onto a cube around the sphere. It is segmented into pieces for each of the six faces of the cube it covers. Then these pieces are sorted by the angle of their 2d projection on the according face of the cube. Afterwards, the pieces are re-connected to get a closed boundary for each cube face. If the part outside the face encloses a corner of the cube, an additional boundary point at this corner is inserted.

The boundaries for each of the faces are triangulated using the same tessellation (OpenGL Architecture Review Board, 1992) as used for planes. This tessellation provides a minimal coarse triangulation. For rendering a finer triangulation is generated using a subdivision step. Each triangle is divided into 25 smaller triangles as depicted in Figure 4.14. The vertices of these small triangles are then projected back onto the surface of the sphere and are rendered.

Figure 4.15 (left) shows an example covering several special cases of the boundary rendering on a sphere. Small triangular boundaries are placed directly at the corners of the projection cube. The large fork shaped boundary crosses an edge of the cube multiple times. All other faces of the cube contain a diamond shaped boundary with points at the middle of the edges of the cube.

Another special case are faces of the cube that are completely surrounded by a boundary but contain no boundary points. They are detected from the on-line construction process of the boundary. Whenever a boundary covers a part of the

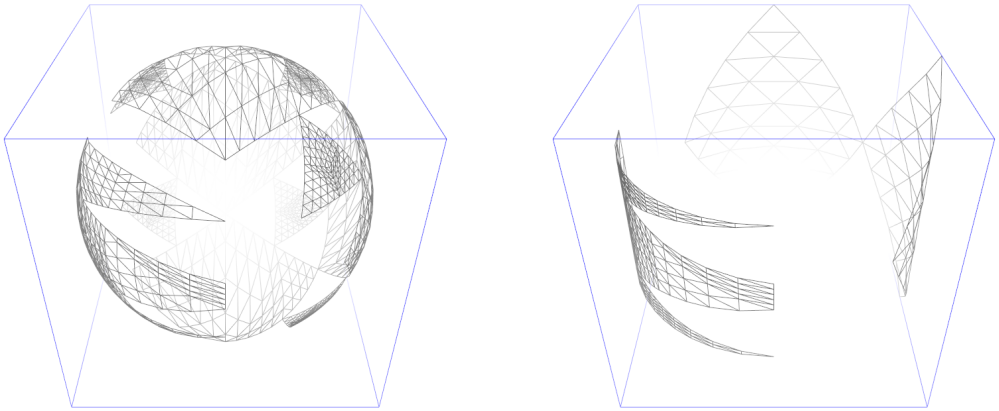


Figure 4.15: Example primitives demonstrating special cases of boundary rendering.

face and the outside part surrounds two or more corners of the projection cube, this face of the cube is marked as filled. If later it does not contain any boundary points, it is rendered as a visible part of the sphere.

Cylinders

Cylindric segments are rendered similar to spherical ones. Instead of six segments on the faces of a cube, the cylinder mantle is projected to only four faces. Again, the boundaries are split to the different faces and then re-connected separately. The closed boundary pieces are then rendered using subdivision and a projection to the cylinder mantle. Figure 4.15 (right) shows some of the same boundaries as on the sphere projected to a cylinder. In difference to the sphere, the cylinder has less special cases because there are no corners of three cube faces.

4.9 Results

In this Section, the effectiveness of the accumulated means segmentation and reconstruction is demonstrated. It is shown that the method is robust to noise and performs well with real scan data. The method executes at interactive speed and the segmentation process uses little processing power. All results in this Section are computed using the thresholds in Table 4.5. The thresholds are scale invariant because they are either angles or distances relative to radii.

4.9.1 Synthetic scan data

To demonstrate the robustness to noise, synthetic scan data is used, where the true segment type is known, see Section 3.2.

Symbol	Threshold	Value
ε_{sd}	Distance to surface	0.14
ω_{nd}	Normal distance	0.8
ω_{cyl}	Cylinder radius difference	0.1
ω_{sph}	Sphere radius difference	0.1
ε_{nds}	Normal distance between $\tilde{\boldsymbol{p}}$	0.4
$\varepsilon_{\sphericalangle}$	Angle between $\tilde{\boldsymbol{n}}$	20°
ε_{ca}	Angle between $\tilde{\boldsymbol{d}}_{\Delta}$	20°
ε_{cr}	Difference between $\bar{R}_{\delta}$	0.2
ε_{cc}	Deviation of cylinder axes	0.4
ε_{sr}	Difference between $\bar{R}_{s}$	0.2
ε_{sc}	Distance of $\bar{\boldsymbol{p}}_{s}$	0.4

Table 4.5: Scale invariant thresholds used for segmentation and reconstruction with accumulated means.

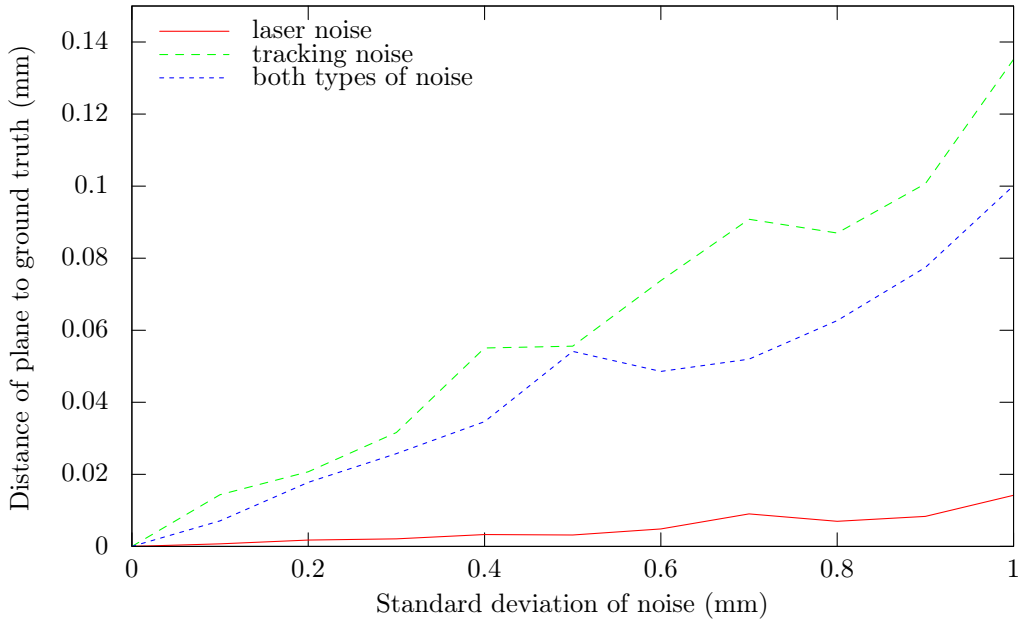


Figure 4.16: Plot of the average normal distance of 25 experiments for noise levels $\sigma = i/10[\text{mm}], i = 0, \ldots, 10$, for plane reconstruction.

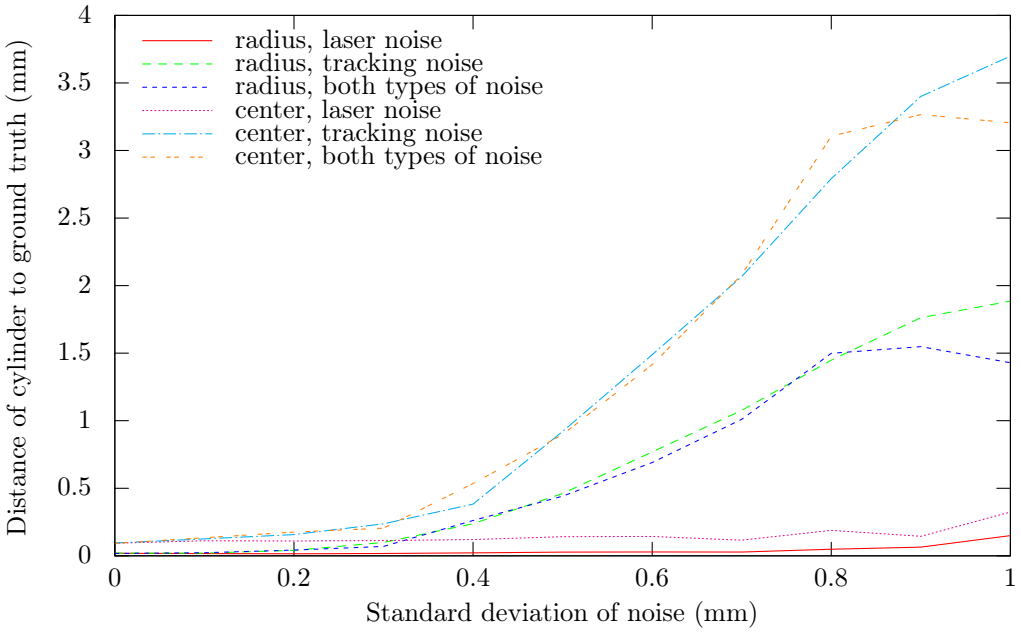


Figure 4.17: Plot of the average radius distance and average distances of the center lines of 25 experiments for noise levels $\sigma = i/10[\text{mm}], i = 0, \ldots, 10$, for cylinder reconstruction.

The plots in Figures 4.16, 4.17, and 4.18 show the results of experiments with synthetic planar, cylindrical, and spherical data. For each level of noise with standard deviation $\sigma = i/10[mm], i = 0, \ldots, 10$, 25 experimental segmentations and reconstructions were computed. For each experiment new noise is generated. The plots show the average distance of the parameters of plane, cylinder, and sphere to the known ground truth parameters. For planes the average normal distance is plotted. For cylinders and spheres the average radius difference and the average distances of the center lines or points are plotted.

The evaluation shows that tracking noise has the largest effect on the reconstruction. The plane reconstruction even improves when laser noise is added to the tracking noise. Cylindrical and spherical reconstructions lead to better results for the radius than for the center line or point. Planar reconstruction is least affected by noise followed by spherical reconstruction. The added noise has the most influence on the cylindrical reconstruction.

The observations follow directly from the used scores. Planar reconstruction is based on positions $\boldsymbol{p}$ and normals $\boldsymbol{n}$ that are quite robust to noise. Spheres also use the mean curvatures H for the radius. The cylinder radius depends on principal curvatures κ_1, κ_2 and principal directions $\boldsymbol{d}_1$, $\boldsymbol{d}_2$. Especially the principal directions are effected substantially by local noise. The reconstruction errors of cylinder radii are around 10 times larger than the errors of sphere reconstruction for the same

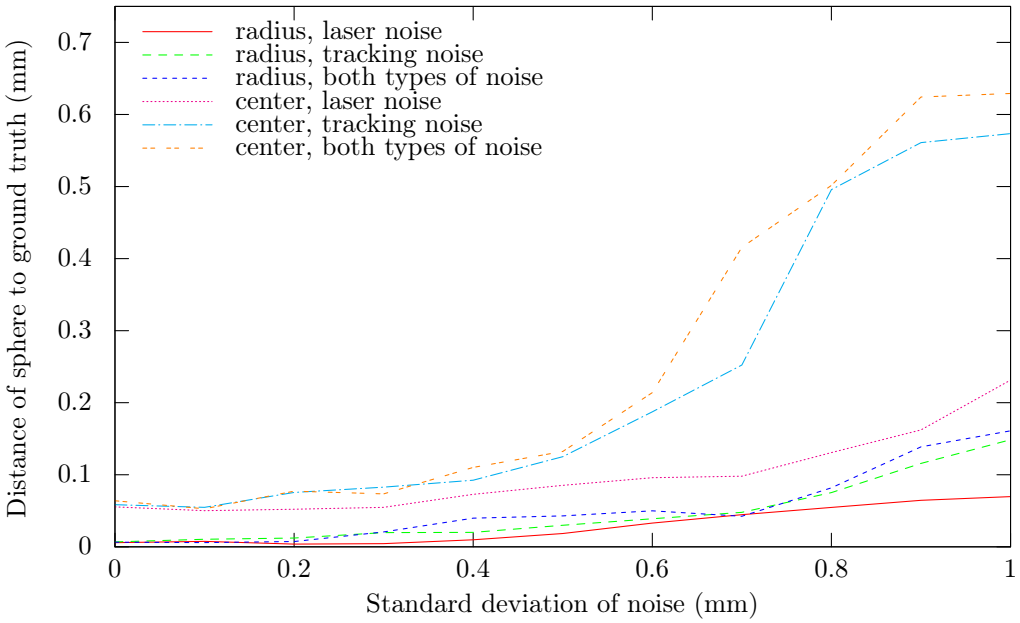


Figure 4.18: Plot of the average radius distance and average distances of the center points of 25 experiments for noise levels $\sigma = i/10[\text{mm}], i = 0, \ldots, 10$, for sphere reconstruction.

levels of tracking noise. Both segment types use radii $\bar{r}_1$, $\bar{r}_2$, $\bar{r}_s$ and normals $\boldsymbol{n}$ for center reconstruction. Here, the error for cylinders is six times larger than the error for sphere in the presence of tracking noise.

While the precision of the reconstruction is affected by the noise none of the reconstructions has failed. There are no outliers with extremely high error rates in the data of these plots. Noise with standard deviation $\sigma > 1$ mm would cause the segments to split into smaller segments, reconstructing parts of the data. Most of these smaller segments would be planar as a curvature based reconstruction would fail.

4.9.2 Hand-tracked synthetic scan data

In Section 3.2.2 synthetic scans were created using real hand movements from a tracking device. The results from these datasets are demonstrated here.

For all synthetic datasets with noise level $\sigma < 0.5$ mm the segmentation produces perfect results: a single segment containing all n-balls is reconstructed.

For the evaluation of these scans, Figure 4.19 is based on the same parameters as in Figures 4.16, 4.17, 4.18 for both types of noise. The non-uniform distribution of the scan lines does not influence the noise robustness for planar reconstruction much. The partial scans of the cylinder and sphere are much more affected by noise.

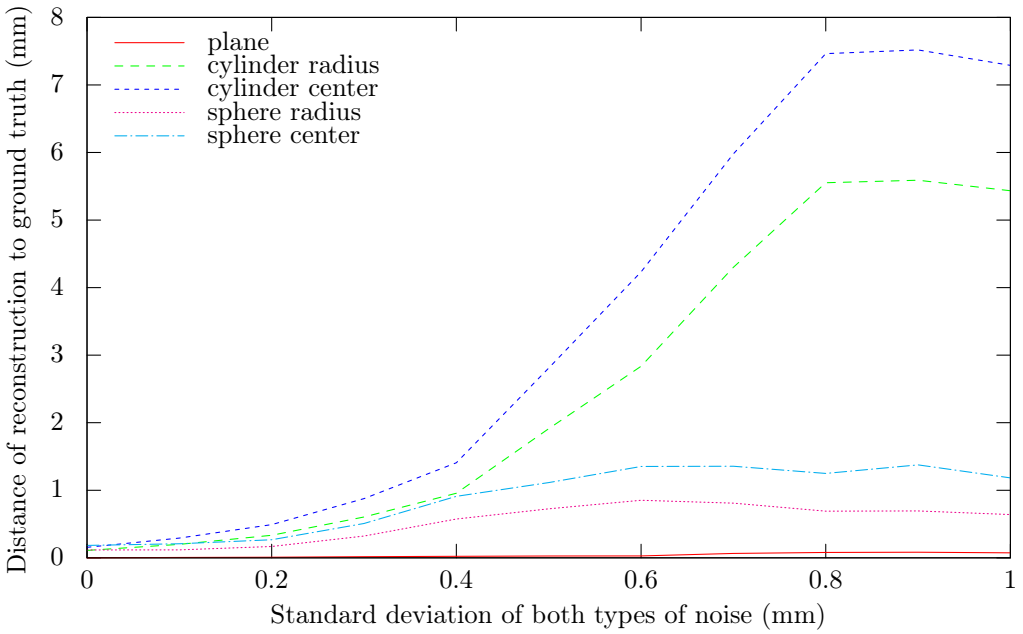


Figure 4.19: Plot of average distances of 25 experiments for noise levels $\sigma = i/10[mm], i = 0, \ldots, 10$, all segment types.

Their reconstructions have errors roughly twice as large as those for the complete dataset. Since only one half of the objects is scanned, the corrected radii R_i, R_s do not have the same stabilizing effect as for complete scans.

4.9.3 Real scans

Tests with real scan data are necessary for evaluation of the on-line reconstruction method. The objects were scanned with a *Faro Edge Laser ScanArm*, see Figure 4.20.

Toy fire truck

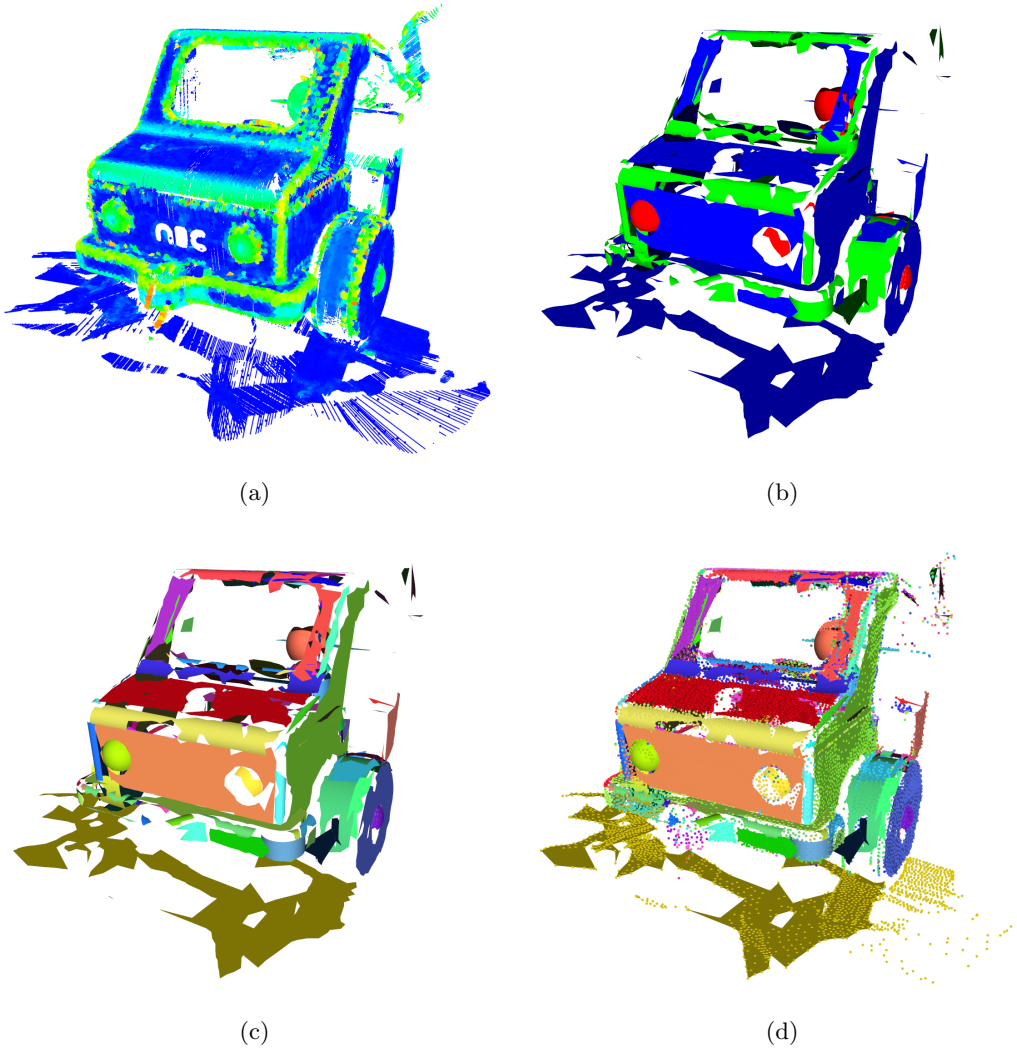
For the experiments presented here, a toy fire truck shown in Figure 4.21 was scanned. It has a length of 0.6 meters and is made from wood with black rubber around the wheels. It's surface consists of many basic geometric primitives. A lot of flat parts can be reconstructed as planes. The rounded edges, the wheels, and the holes at the ladder should be reconstructed as cylinders. Spherical regions are the front lights, the center of the wheels, and the head of the simple driver figure.

Scan of the toy fire truck's cabin

The first scan captures the driver's cabin of the toy fire truck. The simple driver figure was placed behind the steering wheel. Figure 4.22 shows that most of the

Figure 4.20: The Faro Edge ScanArm, a measurement arm based hand-held laser scanner used to capture the data for the results of the on-line reconstruction with accumulated means.

Figure 4.21: Photograph of a wooden fire truck toy.



(a) (b)

(c) (d)

Figure 4.22: Raw data and reconstruction from a scan of the driver's cabin of the wooden fire truck toy: 1 271 342 points, 12 325 scan lines, 15 013 n-balls, 1 011 segments. Raw data colored by curvature magnitude (a), segments colored by segment type: planar, cylindrical, and spherical (b), or each segment individually (c), additionally the n-ball vertices are displayed in the same colors in (d).

segments were detected correctly. The black rubber wheels have a lower scanning quality and are not detected as single cylindrical segments. Instead they are split to multiple cylindrical and flat segments. The wooden side of the wheel is classified correctly into a flat segment and the spherical segment at the axis. All rounded corners of the driver's cabin are reconstructed as cylinders with small radii. At the front left corner of the hood, a small spherical segment can be seen. The flat area at the radiator grill is reconstructed correctly and has two holes for the spheres representing the lights.

Some problems with the boundary reconstruction can be clearly seen. The shape of the boundaries looks incomplete and jagged. In Figure 4.22d it can be seen that the n-ball vertices are covering the missing regions, but the boundary does not enclose them correctly. Especially at the surface below the toy truck a bigger area of the segment is missing from the boundary.

Scan of the toy fire truck's ladder

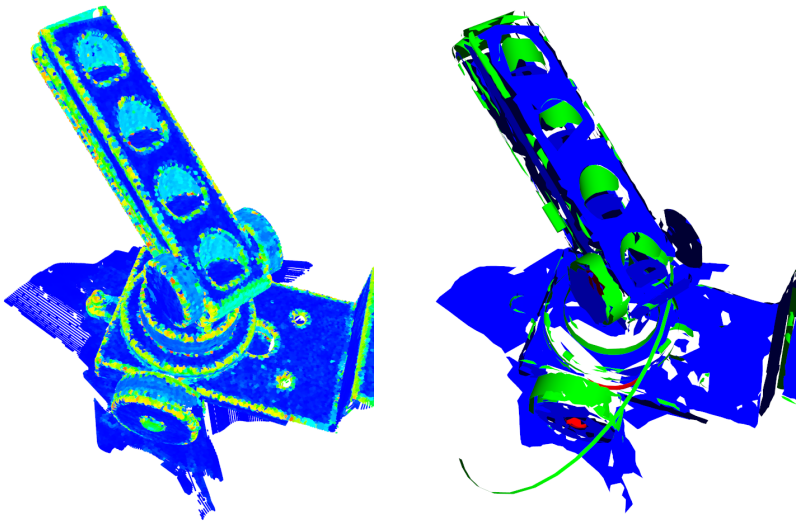
The second scan shown in Figure 4.23 captures the rear end of the toy fire truck including the ladder. Big cylinders at the bottom of the ladder platform show the capabilities of the accumulated means reconstruction to correctly classify segments with large radii. The ladder consists of four cylindrical holes. They are scanned from the inside and are reconstructed and rendered correctly as cylinders with surface normals to the inside. During the accumulated means calculations, the sign of the principal curvatures is preserved and used as a sign for the radius. Then this radius sign is used during rendering to determine the normal orientation of the segment surface. The right rear wheel looks better than the front wheel visible in the last scan. A single cylinder mantle is reconstructed for the black rubber tire. Another one with slightly smaller radius is reconstructed for the narrow stripe of the wood wheel that is visible below the tire. In Figures 4.23c and d the tire is colored purple, while the cylinder mantle of the wood wheel is colored blue.

A typical error of the boundary rendering can be seen as a narrow green cylinder in Figures 4.23b to d. Parts of the cylinder temporarily contained boundary points and are later assumed to be inside the boundary. Therefore, almost the complete cylinder is rendered instead of only a small area inside the boundary.

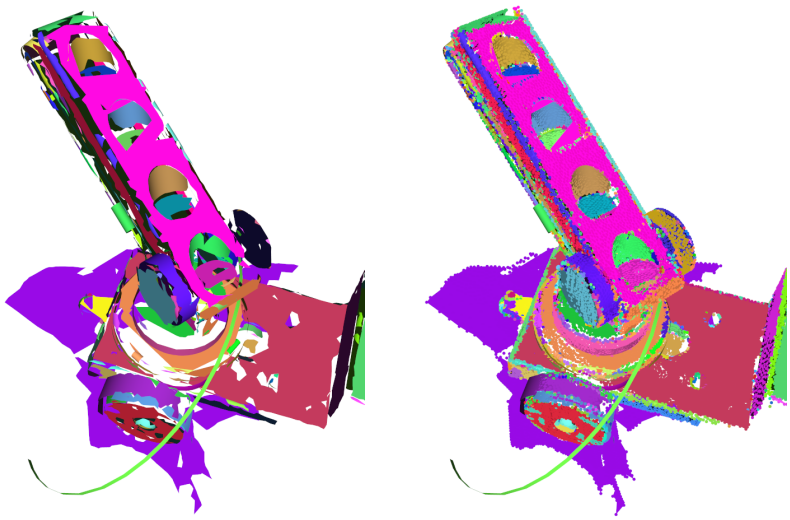
4.9.4 Boundary reconstruction issues

While the segmentation and reconstruction of the basic geometric primitives work quite robust, the boundary reconstruction has several issues. Still, it is better to display the results with the boundaries. Otherwise, it would be necessary to always display the full geometric primitives. This would create a lot of overlapping geometry. It would not be possible to see the correct shape of the reconstructed geometry.

Most problems with the on-line boundary reconstruction are caused by the update-behaviour of the ball tree, see Section 4.1. Whenever an n-ball contains too many



(a) (b)



(c) (d)

Figure 4.23: Raw data and reconstruction from a scan of the ladder of the wooden fire truck toy: 3 128 055 points, 26 749 scan lines, 32 002 n-balls, 1 733 segments. Raw data colored by curvature magnitude (a), segments colored by segment type: planar, cylindrical, and spherical (b), or each segment individually (c), additionally the n-ball vertices are displayed in the same colors in (d).

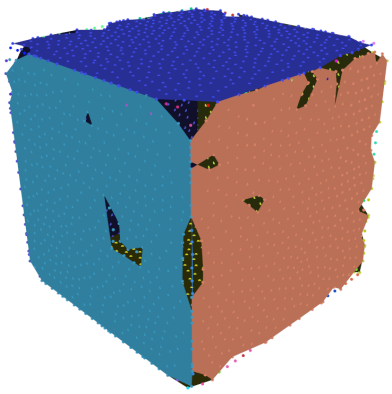


Figure 4.24: Reconstruction of the synthetic scan of a perfect cube, each segment colored individually.

raw-points and is subdivided into smaller n-balls, a small hole is created and filled. This causes a gap in the boundary to appear and disappear again. The on-line boundary reconstruction not always succeeds in finding the right replacement for the removed n-balls. This can be seen at the synthetic scan of a cube in Figure 4.24. Just before the scan is finished, most n-balls are replaced by smaller n-balls. Because many n-balls are replaced almost simultaneously during the on-line reconstruction process, this causes big problems with the boundaries. Narrow gaps toward the inside are introduced. Sometimes they are closed at the outside first and holes are created, as can be seen on the light blue cube face on the left. In other rare cases, even intersections can be introduced to the boundary. Once, the boundary contains intersections, they can not always be solved robustly.

Figure 4.24 also shows some problems with the sharp feature detection, see Section 4.5. Small cylindrical and spherical segments are reconstructed at the edges and corners of the cube. The sharp feature detection should prevent this but there are still some areas where the sharp feature is not detected properly.

4.9.5 Performance

While in some of these scans large number of segments were reconstructed, the implementation always performs at interactive speed on an Intel Core 2 Quad Q6600 2.4 GHz computer with 8 GB of RAM. Most of the processing power is used for the ball tree data structure. Especially the iterative least-squares fits for sharp feature detection (see Section 4.5) require much computation time. The thread processing the segmentation and the computation of accumulated means uses 18% on average and 28% at maximum of one processor core.

4.10 Conclusion

This Chapter presented methods for on-line segmentation and reconstruction of CAD geometry from a stream of point data. An advantage of the on-line processing is, that the results can be evaluated immediately by the operator of the hand-held laser scanner.

First a method strictly divided into segmentation and reconstruction was presented, see Sections 4.2 and 4.3. The segmentation is using a region growing approach based purely on local geometry. Quadrics are used as an intermediate representation. Their canonical form allows an easy computation of the parameters of the reconstructed geometric primitives.

A method integrating segmentation and reconstruction in a single step was presented afterwards, see Sections 4.6 and 4.7. The method is based on accumulated means that are used for on-line computation of means of geometric properties. This approach is limited to planar, cylindrical, and spherical shapes. It could be extended to other geometric primitives like cones, ellipsoids, tori, and rolling ball blends.

The on-line reconstruction of boundaries was described in Section 4.8. There are still some issues with the presented method that were addressed in Section 4.9.4. The robustness of the boundary reconstruction is not as good as for the on-line reconstruction itself.

A possible solution to the problems with the boundary reconstruction would be to include a on-line triangulation similar to Denker et al. (2008, 2011) to the on-line reconstruction. It would be necessary to modify it to allow separate triangulations for each segment. Detecting and updating the boundary of these triangulated segments should be easier and more robust than the method currently used.

Another direction into which the on-line reconstruction could be extended is the processing of dense point data. Instead of reducing the point stream with the ball-tree algorithm, local geometric properties could be calculated for each point in the data from the scanner. A simplified more efficient computation of the principal curvatures could be used, that does not need to fit a local surface. Also, an efficient implementation with support of hardware acceleration techniques would be necessary to achieve this goal.

This Chapter only discussed the reconstruction of regular geometry and basic geometric primitives. CAD models usually consist of a mixture of such regular shapes and free-form geometry. A on-line reconstruction of free-form geometry could be very useful in this context.

Bibliography

G. Agin and T. Binford. Computer description of curved objects. *IEEE Trans. on Computers*, C-25(4):439–449, 1976.

C. Bender, K. Denker, M. Friedrich, K. Hirt, and G. Umlauf. A hand-held laser scanner based on multi-camera stereo-matching. In C. Garth, A. Middel, and H. Hagen, editors, *VLUDS 2011*, volume 27 of *OpenAccess Series in Informatics (OASIcs)*, pages 123–133, 2012.

P. Benkő and T. Várady. Direct segmentation of smooth, multiple point regions. In *Geometric Modeling and Processing*, pages 169–178. IEEE, 2002.

P. Benkő, R. R. Martin, and T. Várady. Algorithms for reverse engineering boundary representation models. *Computer-Aided Design*, 33(11):839–851, 2001.

P. J. Besl and R. C. Jain. Segmentation through variable-order surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(2):167–192, 1988.

G. Biegelbauer and M. Vincze. Efficient 3d object detection by fitting superquadrics to range image data for robot's object manipulation. In *International Conference on Robotics and Automation*, pages 1086–1091. IEEE, 2007.

T. Bodenmüller and G. Hirzinger. Online surface reconstruction from unorganized 3d-points for the DLR hand-guided scanner system. In *2nd Symp. on 3D Data Processing, Visualization and Transmission*, pages 285–292, 2004.

J. Boesch and R. Pajarola. Flexible configurable stream processing of point data. In *WSCG'2009*, pages 49–56, 2009.

J. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

N. D. Campbell, G. Vogiatzis, C. Hernández, and R. Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *Proc. of the 10th European Conf. on Computer Vision*, pages 766–779, 2008.

M. Caputo, K. Denker, B. Dums, and G. Umlauf. 3D hand gesture recognition based on sensor fusion of commodity hardware. In H. Reiterer and O. Deussen, editors, *Mensch & Computer 2012*, pages 293–302, 2012.

T. Chan, G. Golub, and R. Leveque. Algorithms for computing the sample variance: analysis and recommendations. *The American Statistician*, 37(3):242–247, 1983.

J.-C. Cheng and S.-J. Feng. A real-time multiresolutional stereo matching algorithm. In *ICIP (3)*, pages 373–376, 2005.

S. P. Clode, E. E. Zelniker, P. J. Kootsookos, and I. V. L. Clarkson. A phase-coded disk approach to thick curvilinear line detection. In *Proceedings of the 12th European Signal Processing Conference*, pages 1147–1150, 2004.

K. Denker and G. Umlauf. Accurate real-time multi-camera stereo-matching on the gpu for 3d reconstruction. *Journal of WSCG*, 19(1):9–16, 2011a.

K. Denker and G. Umlauf. Survey on benchmarks for a gpu based multi camera stereo matching algorithm. In A. Middel, I. Scheler, and H. Hagen, editors, *VLUDS 2010*, volume 19 of *OpenAccess Series in Informatics (OASIcs)*, pages 20–26, 2011b.

K. Denker, B. Lehner, and G. Umlauf. Online triangulation of laser-scan data. In R. Garimella, editor, *17th International Meshing Roundtable*, pages 415–432, 2008.

K. Denker, B. Lehner, and G. Umlauf. Real-time triangulation of point streams. *Engineering with Computers*, 27(1):67–80, 2011.

K. Denker, D. Hagel, J. Raible, G. Umlauf, and B. Hamann. On-line reconstruction of cad geometry. In *International Conference on 3D Vision*, pages 151–158, 2013.

F. Devernay. A non-maxima suppression method for edge detection with sub-pixel accuracy. Technical Report RR-2724, INRIA, 1995.

F. Devernay and O. Faugeras. Straight lines have to be straight: automatic calibration and removal of distortion from scenes of structured enviroments. *Mach. Vision Appl.*, 13(1):14–24, 2001.

G. Frankowski, M. Chen, and T. Huth. Real-time 3d shape measurement with digital stripe projection by texas instruments micro mirror devices DMD$^{\mathrm{TM}}$. *Three-Dimensional Image Capture and Applications III*, 3958(1):90–105, 2000.

Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. In *CVPR*, pages 1–8, 2007.

I. Gronau and S. Moran. Optimal implementations of upgma and other common clustering algorithms. *Information Processing Letters*, 104(6):205–210, 2007.

A. W. Gruen. Adaptive least squares correlation: A powerful image matching technique. *South African Journal of Photogrammetry, Remote Sensing and Cartography*, 14:175–187, 1985.

H. Hattori and A. Maki. Stereo matching with direct surface orientation recovery. In *In Ninth British Machine Vision Conference*, pages 356–366, 1998.

J. Hensler, K. Denker, M. O. Franz, and G. Umlauf. Hybrid face recognition based on real-time multi-camera stereo-matching. In G. Bebis et al., editors, *Advances in Visual Computing*, volume 6939 of *Lecture Notes in Computer Science*, pages 158–167, 2011.

P. W. Holland and R. E. Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics: Theory and Methods*, A6:813–827, 1977.

A. Hornung and L. Kobbelt. Interactive pixel-accurate free viewpoint rendering from images with silhouette aware sampling. *Comp. Graph. Forum*, 28(8):2090–2103, 2009.

S. Inokuchi, K. Sato, and F. Matsuda. Range-imaging for 3-d object recognition. In *International Conference on Pattern Recognition*, pages 806–808, 1984.

R. A. Jarvis. A perspective on range finding techniques for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):122–139, 1983a.

R. A. Jarvis. A laser time-of-flight range scanner for robotic vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 5(5):505–512, 1983b.

A. Klaus, M. Sormann, and K. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *Proc. of the 18th Int. Conf. on Pattern Recognition*, pages 15–18, 2006.

D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*, chapter 4.2.2, page 232. Addison-Wesley, third edition, 1998.

R. Koch, M. Pollefeys, and L. V. Gool. Realistic 3-d scene modeling from uncalibrated image sequences. In *ICIP'99, Kobe: Japan*, pages 500–504, 1999.

V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1568–1583, 2006.

C. Lei, J. Selzer, and Y.-H. Yang. Region-tree based stereo using dynamic programming optimization. In *Proc. of the 2006 IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2378–2385, 2006.

M. Levoy. The stanford spherical gantry. `http://graphics.stanford.edu/projects/gantry/`, 2002.

J. P. Lewis. Fast template matching. In *Vision Interface*, pages 120–123, 1995.

R. F. Ling. Comparison of several algorithms for computing sample means and variances. *Journal of the American Statistical Association*, 69(348):859–866, 1974.

K. Moreland and E. Angel. The FFT on a GPU. In *Proc. of the ACM Conf. on Graphics Hardware*, pages 112–119, 2003.

D. Murray and J. Little. Using real-time stereo vision for mobile robot navigation. In *Autonomous Robots*, pages 161–171, 2000.

Y. Nakamura, T. Matsuura, K. Satoh, and Y. Ohta. Occlusion detectable stereo – occlusion patterns in camera matrix. In *CVPR*, pages 371–378, 1996.

OpenGL Architecture Review Board. *OpenGL reference manual: the official reference document for OpenGL, release 1.* Otl Series. Addison-Wesley, 1992.

D. Page, Y. Sun, A. Koschan, J. Paik, and M. Abidi. Normal vector voting: Crease detection and curvature estimation on large, noisy meshes. *Graph. Models*, 64: 199–229, 2002.

R. Pajarola. Stream-processing points. In *IEEE Visualization*, pages 239–246, 2005.

D. Pham and L. Hieu. Reverse engineering – hardware and software. In V. Raja and K. Fernandes, editors, *Reverse Engineering – An Industrial Perspective*, pages 33–30, 2008.

J. L. Posdamer and M. D. Altschuler. Surface measurement by space-encoded projected beam systems. *Computer Graphics and Image Processing*, 18(1):1–17, 1982.

W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, 2007.

Qt Project Hosting. Qt documentation. `http://qt-project.org/doc/qt-4.8/containers.html\#algorithmic-complexity`, 2013.

S. Rusinkiewicz and M. Levoy. Qsplat: A multiresolution point rendering system for large meshes. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 343–352, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3d model acquisition. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 438–446, 2002.

A. M. Sá, E. S. de Medeiros Filho, P. C. P. Carvalho, and L. Velho. Coded structured light for 3d-photography: An overview. *RITA*, 9(2):203–219, 2002.

H. Sadeghi, P. Moallem, and S. A. Monadjemi. Feature based dense stereo matching using dynamic programming and color. *International Journal of Computational Intelligence*, 4(3):179–186, 2008.

G. Sansoni, S. Corini, S. Lazzari, R. Rodella, and F. Docchio. Three-dimensional imaging based on gray-code light projection: characterization of the measuring algorithm and development of a measuring system for industrial applications. *Applied Optics*, 36(19):4463–4472, 1997.

D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1–3):7–42, 2002.

D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *Proceedings of IEEE CVPR 2003*, pages 195–202, 2003.

D. Scharstein and R. Szeliski. Middlebury stereo vision page. `http://vision.middlebury.edu/stereo/`, 2007.

R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, 2007.

S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. of the 2006 IEEE Conf. on Computer Vision and Pattern Recognition*, pages 519–528, 2006.

S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. Multi-view stereo. `http://vision.middlebury.edu/mview/`, 2009.

A. Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.

T. Shibahara, T. Aoki, H. Nakajima, and K. Kobayashi. A sub-pixel stereo correspondence technique based on 1d phase-only correlation. In *ICIP07*, pages 221–224, 2007.

Y. Shirai and S. Tsuji. Extraction of the line drawing of 3-dimensional objects by sequential illumination from several directions. *Pattern Recognition*, 4(4):343–351, 1972.

Sixense. Razer hydra motion sensing controller. `http://sixense.com/hardware/razerhydra`, 2014.

R. R. Sokal and C. D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin*, 28:1409–1438, 1958.

C. Strecha. Multi-view stereo test images. `http://cvlab.epfl.ch/~strecha/multiview/`, 2008.

C. Strecha, W. von Hansen, L. J. V. Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *CVPR*, pages 1–8, 2008.

J. Tajima and M. Iwakawa. 3-d data acquisition by rainbow range finder. In *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, volume i, pages 309–313 vol.1, Jun 1990.

C. Teutsch. *Model-based Analysis and Evaluation of Point Sets from Optical 3D Laser Scanners*. Shaker Verlag, 2007.

A. J. van Reeken. Letters to the editor: Dealing with Neely's algorithms. *Commun. ACM*, 11(3):149–150, 1968.

M. Vančo, B. Hamann, and G. Brunnett. Surface reconstruction from unorganized point data with quadrics. *Computer Graphics Forum*, 27(6):1593–1606, 2008.

T. Várady. Reverse engineering of geometric models – an introduction. *Comp.-Aided Design*, 29(4):255–268, 1997.

F. Voltolini, S. El-Hakim, F. Remondino, and L. Gonzo. Effective high resolution 3d geometric reconstruction of heritage and archaeological sites from images. In *Proc. of the 35th CAA Conference*, pages 43–50, 2007.

P. Vuylsteke and A. Oosterlinck. Range image acquisition with a single binary-encoded light pattern. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(2):148–164, 1990.

J. Wang and Z. Yu. Surface feature based mesh segmentation. *Computers and Graphics*, 35(3):661–667, 2011.

C. Weber, S. Hahmann, and H. Hagen. Sharp feature detection in point clouds. In *Proceedings of the 2010 Shape Modeling International Conference*, SMI '10, pages 175–186, Washington, DC, USA, 2010. IEEE Computer Society.

A. Wehr and U. Lohr. Airborne laser scanning – an introduction and overview. *ISPRS Journal of Photogrammetry & Remote Sensing*, 54(2–3):68–82, 1999.

B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

S. Winkelbach, S. Molkenstruck, and F. M. Wahl. Low-cost laser range scanner and fast surface registration approach. In *DAGM-Symposium*, pages 718–728, 2006.

J. Wu and L. Kobbelt. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum*, 24(3):277–284, 2005.

I. Yamazaki, V. Natarajan, Z. Bai, and B. Hamann. Segmenting point-sampled surfaces. *The Visual Computer*, 26(12):1421–1433, 2010.

R. Yang and M. Pollefeys. A versatile stereo implementation on commodity graphics hardware. *Real-Time Imaging*, 11(1):7–18, 2005.

R. Yang, G. Welch, and G. Bishop. Real-time consensus-based scene reconstruction using commodity graphics hardware. In *Proc. of the 10th Pacific Conf. on Computer Graphics and Applications*, pages 225–235, 2002.

C. Zach, M. Sormann, and K. F. Karner. High-performance multi-view reconstruction. In *3DPVT*, pages 113–120, 2006.

C. Zhang and T. Chen. A self-reconfigurable camera array. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Sketches*, page 151, 2004.

S. Zhang and S.-T. Yau. High-resolution, real-time 3d absolute coordinate measurement based on a phase-shifting method. *Opt. Express*, 14(7):2644–2649, 2006.

S. Zhong. Efficient online spherical k-means clustering. In *Proc. 2005 IEEE International Joint Conference on Neural Networks*, volume 5, pages 3180–3185, 2005.

Curriculum Vitæ – Klaus Denker

Studies and Career

April 2002 to January 2008	**Diploma Studies in Computer Science** University of Kaiserslautern
July 2008 to August 2009	**Wissenschaftlicher Mitarbeiter** Dept. of Computer Science, University of Kaiserslautern
September 2009 to December 2013	**Wissenschaftlicher Mitarbeiter** Dept. of Computer Science, HTWG Konstanz
January 2014 to March 2014	**Doctoral Stipend** University of Kaiserslautern
since April 2014	**Software Developer** Dassault Systèmes, Stuttgart, Germany

Academic Degree

January 2008	**Diploma in Computer Science (Dipl.-Inf.)** University of Kaiserslautern

Awards

November 2009	**Zukunftspreis Pfalz 2009, Nachwuchspreis** Bezirksverband Pfalz

Publication List

K. Denker, B. Lehner, and G. Umlauf. Online triangulation of laser-scan data. In R. Garimella, editor, *17th International Meshing Roundtable*, pages 415–432, 2008.

K. Denker, B. Lehner, and G. Umlauf. Real-time triangulation of point streams. *Engineering with Computers*, 27(1):67–80, 2011.

K. Denker and G. Umlauf. Accurate real-time multi-camera stereo-matching on the gpu for 3d reconstruction. *Journal of WSCG*, 19(1):9–16, 2011a.

K. Denker and G. Umlauf. Survey on benchmarks for a gpu based multi camera stereo matching algorithm. In A. Middel, I. Scheler, and H. Hagen, editors, *VLUDS 2010*, volume 19 of *OpenAccess Series in Informatics (OASIcs)*, pages 20–26, 2011b.

J. Hensler, K. Denker, M. O. Franz, and G. Umlauf. Hybrid face recognition based on real-time multi-camera stereo-matching. In G. Bebis et al., editors, *Advances in Visual Computing*, volume 6939 of *Lecture Notes in Computer Science*, pages 158–167, 2011.

C. Bender, K. Denker, M. Friedrich, K. Hirt, and G. Umlauf. A hand-held laser scanner based on multi-camera stereo-matching. In C. Garth, A. Middel, and H. Hagen, editors, *VLUDS 2011*, volume 27 of *OpenAccess Series in Informatics (OASIcs)*, pages 123–133, 2012.

M. Caputo, K. Denker, B. Dums, and G. Umlauf. 3D hand gesture recognition based on sensor fusion of commodity hardware. In H. Reiterer and O. Deussen, editors, *Mensch & Computer 2012*, pages 293–302, 2012.

K. Denker, D. Hagel, J. Raible, G. Umlauf, and B. Hamann. On-line reconstruction of cad geometry. In *International Conference on 3D Vision*, pages 151–158, 2013.